Package: medrobust (via r-universe)

October 26, 2026

Type Package

Title Robust Causal Mediation Analysis Under Differential Misclassification

Version 0.4.4

Maintainer Davood Tofighi <dtofighi@gmail.com>

Description Provides tools for conducting sensitivity analysis for causal mediation effects when the exposure or mediator is measured with differential misclassification (e.g., recall bias, outcome-dependent measurement error). Unlike existing measurement error correction methods that assume non-differential error or require validation data, 'medrobust' derives partial identification bounds that remain valid without gold-standard measurements. The package implements methods developed in Tofighi (2025) for partial identification bounds for Natural Direct Effects (NDE) and Natural Indirect Effects (NIE), data-driven falsification via testable implications, sensitivity analysis over user-specified ranges of misclassification parameters, diagnostic tools and publication-quality visualizations, bootstrap inference for confidence intervals via percentile and bias-corrected and accelerated (BCa) methods, and synthetic data generation for power analysis and methods research. The package handles both mediator misclassification and exposure misclassification within a unified framework. The partial identification approach builds on Manski (2003, ISBN:978-0387004549); confidence intervals for the partially identified bounds use the construction of Imbens and Manski (2004) <doi:10.1111/j.1468-0262.2004.00549.x>.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

Depends R (>= 4.1.0)

Imports S7 (>= 0.1.0), dplyr (>= 1.1.0), ggplot2 (>= 3.4.0), stats, utils, rlang, parallel

Suggests testthat (>= 3.0.0), knitr, rmarkdown, foreach, doParallel, boot, data.table, nhanesA, mediation (>= 4.5.0), EValue (>= 4.1.0), covr, gridExtra, scales

Config/testthat/edition 3

URL https://github.com/data-wise/medrobust, https://data-wise.github.io/medrobust/

BugReports https://github.com/data-wise/medrobust/issues

Config/roxygen2/version 8.1.0

Repository https://data-wise.r-universe.dev

Date/Publication 2026-09-26 05:53:41 UTC

RemoteUrl https://github.com/Data-Wise/medrobust

RemoteRef main

RemoteSha 61d0afb9f1af996887c0f65b8319ceb29d760af4

Contents

as.data.frame.medrobust_bounds . . . 3
as.list.sensitivity_region . . . 3
as_sensitivity_region . . . 4
bootstrap_results . . . 5
bootstrap_width_summary . . . 6
bound_ci . . . 6
bound_ne . . . 8
check_compatibility . . . 12
compare_bounds . . . 15
compatibility_test . . . 16
compute_bound_se . . . 17
extract_bounds . . . 17
extract_falsified_region . . . 18
falsification_summary . . . 19
format_effect . . . 21
gesthtn . . . 21
heals_data . . . 23
medrobust_bounds . . . 24
new_falsification_summary . . . 25
nhanes_pa . . . 27
plot_bootstrap_distribution . . . 28
power_analysis . . . 29
power_analysis_result . . . 31
print.compatibility_test . . . 32
sensitivity_plot . . . 32
sensitivity_region . . . 34

simulate_dm_data 35
simulated_dm_data 37
summary.compatibility_test 38
test_multiple_hypotheses 38

Index **40**

`as.data.frame.medrobust_bounds` *Convert partial identification bounds to a data frame*

Description

`as.data.frame()` on a `medrobust_bounds` object (as returned by `bound_ne`) returns a one-row data frame of the bounds and fit summary, convenient for tabulating or combining results across analyses.

This is an S7 method for the base generic; call it as `as.data.frame(bounds)`.

Arguments

`x`	A `medrobust_bounds` object.
`...`	Ignored.

Value

A one-row `data.frame` with columns `NIE_lower`, `NIE_upper`, `NDE_lower`, `NDE_upper`, `NIE_width`, `NDE_width` (upper minus lower), `effect_scale`, `misclassified_variable`, `n_compatible`, `n_evaluated`, and `falsified_proportion`. If the object carries bootstrap results, it also has percentile-interval columns for each bound endpoint (`NIE_lower_ci_lower`, `NIE_lower_ci_upper`, ..., `NDE_upper_ci_upper`) plus `bootstrap_method` and `bootstrap_n_reps`. Analytic (Imbens-Manski) intervals are not included; use `x@analytic_ci`.

See Also

`bound_ne`, `medrobust_bounds`

`as.list.sensitivity_region` *Convert a sensitivity region to a list*

Description

`as.list()` on a sensitivity region object (as returned by `sensitivity_region`) returns its four parameter ranges as a plain named list, the same shape accepted by the `sensitivity_region` argument of `bound_ne`.

This is an S7 method for the base generic; call it as `as.list(region)`.

Arguments

x	A sensitivity region object created by sensitivity_region.
...	Ignored.

Value

A named list with elements sn0_range, sp0_range, psi_sn_range, and psi_sp_range, each a length-2 numeric vector c(lower, upper).

See Also

sensitivity_region, bound_ne

---

as_sensitivity_region *Create sensitivity_region object from list*

Description

Create sensitivity_region object from list

Usage

```
as_sensitivity_region(region_list)
```

Arguments

region_list	List with sn0_range, sp0_range, psi_sn_range, psi_sp_range

Value

sensitivity_region S7 object

Examples

```
as_sensitivity_region(list(
  sn0_range = c(0.80, 0.95), sp0_range = c(0.85, 0.99),
  psi_sn_range = c(1, 2), psi_sp_range = c(1, 1)
))
```

`bootstrap_results` *Bootstrap Results Class*

Description

Stores bootstrap inference results for partial identification bounds.

Usage

```
bootstrap_results(
  method = character(0),
  n_reps = integer(0),
  n_failed = 0L,
  confidence_level = integer(0),
  nie_lower_ci = integer(0),
  nie_upper_ci = integer(0),
  nde_lower_ci = integer(0),
  nde_upper_ci = integer(0),
  boot_nie_lower = integer(0),
  boot_nie_upper = integer(0),
  boot_nde_lower = integer(0),
  boot_nde_upper = integer(0),
  z0 = NULL,
  acceleration = NULL
)
```

Arguments

`method`	Character: "percentile" or "bca"
`n_reps`	Number of bootstrap replications
`n_failed`	Number of failed bootstrap samples
`confidence_level`	Confidence level (e.g., 0.95)
`nie_lower_ci`	Confidence interval for NIE lower bound
`nie_upper_ci`	Confidence interval for NIE upper bound
`nde_lower_ci`	Confidence interval for NDE lower bound
`nde_upper_ci`	Confidence interval for NDE upper bound
`boot_nie_lower`	Bootstrap samples for NIE lower bound
`boot_nie_upper`	Bootstrap samples for NIE upper bound
`boot_nde_lower`	Bootstrap samples for NDE lower bound
`boot_nde_upper`	Bootstrap samples for NDE upper bound
`z0`	BCa bias-correction parameter
`acceleration`	BCa acceleration parameter

Value

An S7 object of class `bootstrap_results` holding bootstrap confidence intervals and the replicate samples for the lower and upper bound endpoints of the natural direct and indirect effects.

`bootstrap_width_summary` *Compute Width of Bootstrap Distribution*

Description

Compute the mean, median, and range of bound widths from bootstrap samples.

Usage

```
bootstrap_width_summary(bootstrap_results, effect = "NIE")
```

Arguments

`bootstrap_results`	The `bootstrap_results` property of a `bound_ne` fit with `bootstrap = TRUE`, or the list returned by `compute_bootstrap_ci()`
`effect`	Character string: "NIE" or "NDE"

Value

List with width statistics

`bound_ci` *Confidence intervals for partial-identification bounds (Imbens-Manski)*

Description

Computes a confidence interval for the partial-identification set returned by `bound_ne()`. The raw estimated bound $[\hat{L}, \hat{U}]$ is consistent but is *not* a confidence set: when the identified set is narrow relative to the sampling uncertainty of its endpoints, it under-covers the true effect at small samples. `bound_ci()` widens the endpoints by their standard errors using the Imbens & Manski (2004) construction, restoring approximately nominal coverage of the true effect.

Usage

```
bound_ci(
  bounds,
  data,
  exposure,
  mediator,
  outcome,
  confounders,
  misclassified_variable = c("exposure", "mediator"),
  n_boot = 200L,
  level = 0.95,
  seed = NULL
)
```

Arguments

`bounds`	A fitted `medrobust_bounds` object from `bound_ne()`.
`data`	The data frame passed to `bound_ne()`.
`exposure`, `mediator`, `outcome`, `confounders`	Column names, as in `bound_ne()`.
`misclassified_variable`	Either `"exposure"` or `"mediator"`; selects the recovery used to evaluate the effect at a single sensitivity parameter.
`n_boot`	Number of resamples for the endpoint standard errors (default 200).
`level`	Confidence level (default 0.95).
`seed`	Optional integer seed for reproducibility.

Details

Endpoint standard errors are obtained by re-evaluating the effect at the fixed minimizing/maximizing sensitivity parameter on resampled data (one evaluation per resample, with no grid search), which is far cheaper than a full bootstrap of the whole grid.

Value

A named list with elements `NIE` and `NDE`, each a numeric vector with `lower`, `upper` (the point bounds), `se_lower`, `se_upper` (endpoint SEs), and `ci_lower`, `ci_upper` (the Imbens-Manski confidence interval).

References

Imbens, G. W. and Manski, C. F. (2004). Confidence Intervals for Partially Identified Parameters. *Econometrica*, 72(6), 1845-1857.

See Also

`bound_ne()`

Examples

```
sim <- simulate_dm_data(
  n = 2000,
  true_params = list(beta_AM = log(2.5), theta_AY = log(1.5), theta_MY = log(2.5)),
  dm_params = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
  misclass_type = "mediator", confounders = 1, seed = 1
)
bounds <- bound_ne(
  data = sim@observed, exposure = "A", mediator = "M_star", outcome = "Y",
  confounders = "C1", misclassified_variable = "mediator",
  sensitivity_region = list(
    sn0_range = c(0.80, 0.99), sp0_range = c(0.80, 0.99),
    psi_sn_range = c(0.8, 1.5), psi_sp_range = c(0.8, 1.5)
  ),
  n_grid = 10, verbose = FALSE
)

# Widen the raw bounds [L, U] into Imbens-Manski confidence intervals
ci <- bound_ci(
  bounds, data = sim@observed, exposure = "A", mediator = "M_star",
  outcome = "Y", confounders = "C1", misclassified_variable = "mediator",
  n_boot = 50, seed = 1
)
ci$NDE
```

---

bound_ne *Partial Identification Bounds for Natural Effects Under Differential Misclassification*

Description

Computes partial identification bounds for the Natural Direct Effect (NDE) and Natural Indirect Effect (NIE) when either the exposure or mediator is subject to differential misclassification. The method does not require validation data.

Usage

```
bound_ne(
  data,
  exposure,
  mediator,
  outcome,
  confounders = NULL,
  misclassified_variable = c("exposure", "mediator"),
  sensitivity_region = NULL,
  n_grid = 50,
  effect_scale = c("OR", "RR", "RD"),
```

```
  confidence_level = 0.95,
  ci_method = c("none", "analytic"),
  ci_n_boot = 200L,
  bootstrap = FALSE,
  bootstrap_reps = 1000,
  bootstrap_method = c("percentile", "bca"),
  parallel = FALSE,
  n_cores = NULL,
  cache = FALSE,
  cache_dir = NULL,
  verbose = TRUE,
  stratify_by = NULL,
  use_adaptive_grid = TRUE,
  grid_method = c("lhs", "auto", "regular", "adaptive", "sobol", "binary")
)
```

Arguments

`data`	A data frame containing the observed variables.
`exposure`	Character string. Name of the exposure variable (A or A*).
`mediator`	Character string. Name of the mediator variable (M or M*).
`outcome`	Character string. Name of the outcome variable (Y).
`confounders`	Character vector. Names of confounding variables.
`misclassified_variable`	Character string. Either "exposure" or "mediator" to indicate which variable is subject to misclassification.
`sensitivity_region`	A named list defining the sensitivity region Theta_Psi. Should contain: `sn0_range`, `sp0_range`, `psi_sn_range`, `psi_sp_range`. Each element should be a numeric vector of length 2 giving the minimum and maximum values. If NULL, default ranges are used.
`n_grid`	Integer. Number of grid points per parameter dimension. Default is 50.
`effect_scale`	Character string. Scale for reporting effects: "OR" (odds ratio), "RR" (risk ratio), or "RD" (risk difference). Default is "OR".
`confidence_level`	Numeric. Confidence level for confidence intervals. Default is 0.95.
`ci_method`	Character. `"none"` (default) or `"analytic"`. If `"analytic"`, attaches Imbens-Manski confidence intervals (see `bound_ci()`) to the result's `@analytic_ci` slot.
`ci_n_boot`	Integer. Resamples for the analytic CI endpoint SEs. Default 200.
`bootstrap`	Logical. Whether to compute bootstrap confidence intervals. Default is FALSE.
`bootstrap_reps`	Integer. Number of bootstrap replicates if bootstrap=TRUE. Default is 1000.
`bootstrap_method`	Character string. Bootstrap CI method: "percentile" (default) or "bca" (bias-corrected and accelerated). The percentile method is faster and adequate for

most applications. BCa provides second-order accurate intervals but requires jackknife estimation, which is computationally intensive for large datasets.

`parallel` Logical. Whether to use parallel processing. Default is FALSE.

`n_cores` Integer. Number of cores for parallel processing. If NULL, uses detectCores() - 1. Default is NULL.

`cache` Logical. Whether to cache intermediate results. Default is FALSE.

`cache_dir` Character string. Directory for cache files. If NULL, uses temp directory.

`verbose` Logical. Whether to print progress messages. Default is TRUE.

`stratify_by` Character vector. Additional variables to stratify by (advanced use).

`use_adaptive_grid`
Logical. Whether to use adaptive grid refinement for large grids (n_grid >= 20). This dramatically reduces computation time by focusing on compatible regions. Default is TRUE.

`grid_method` Character string specifying grid search algorithm:

- `"lhs"` (default): Latin Hypercube Sampling - space-filling design that reduces evaluations by 99% while maintaining broad coverage. Best for most applications (McKay et al., 1979).
- `"auto"`: Automatically selects best method based on a 16-point probe of the parameter space.
- `"regular"`: Exhaustive regular grid (n_grid^4 evaluations). Use for exact bounds when computational budget allows.
- `"sobol"`: Sobol low-discrepancy sequences (Sobol, 1967). Similar to LHS but better for high-dimensional problems.
- `"adaptive"`: Two-stage coarse-to-fine refinement. Effective when falsification rate is high.
- `"binary"`: Binary search on parameter boundaries. Efficient when compatibility is monotonic in parameters.

Details

This function implements the partial identification approach described in Tofighi, D. (2025). The method derives bounds on causal mediation effects by specifying a plausible range for misclassification parameters and using testable implications to rule out empirically inconsistent values.

The sensitivity region Theta_Psi is defined by four parameters:

- `sn0`: Baseline sensitivity (probability of correct classification when Y=0)
- `sp0`: Baseline specificity
- `psi_sn`: Differential sensitivity parameter (odds ratio)
- `psi_sp`: Differential specificity parameter (odds ratio)

Non-differential misclassification corresponds to psi_sn = psi_sp = 1.

Value

An object of class `medrobust_bounds` containing:

`NIE_lower`	Lower bound for Natural Indirect Effect
`NIE_upper`	Upper bound for Natural Indirect Effect
`NDE_lower`	Lower bound for Natural Direct Effect
`NDE_upper`	Upper bound for Natural Direct Effect
`compatible_sets`	Data frame of compatible parameter sets
`falsified_proportion`	Proportion of sensitivity region falsified
`effect_scale`	Scale used for reporting
`n_evaluated`	Number of parameter sets evaluated
`evaluated_sets`	Every evaluated parameter set with a logical `compatible` column; see `extract_falsified_region`
`n_compatible`	Number of compatible parameter sets
`computation_time`	Time taken for computation
`call`	Original function call
`data_summary`	Summary statistics of the data
`bootstrap_results`	Bootstrap results if bootstrap=TRUE

References

Tofighi, D. (2025). Partial Identification of Causal Mediation Effects Under Differential Misclassification. *Biostatistics*.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2), 239-245.

Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4), 86-112.

See Also

`check_compatibility`, `sensitivity_plot`, `falsification_summary`

Examples

```
# Simulate data with a known mediator-misclassification mechanism
sim <- simulate_dm_data(
  n = 8000,
  true_params = list(beta_AM = log(2.5), theta_AY = log(1.5), theta_MY = log(2.5)),
  dm_params = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
  misclass_type = "mediator", confounders = 1, seed = 1
)
```

```
# Sensitivity region containing the (here non-differential) truth
sens_region <- list(
  sn0_range = c(0.80, 0.99),
  sp0_range = c(0.80, 0.99),
  psi_sn_range = c(0.8, 1.5),
  psi_sp_range = c(0.8, 1.5)
)

# Compute partial-identification bounds for mediator misclassification.
# The raw bound [L, U] is consistent but is NOT a confidence set; at finite n
# it can under-cover the truth, so we add Imbens-Manski confidence intervals
# in the same fit via ci_method = "analytic".
set.seed(1)
bounds <- bound_ne(
  data = sim@observed,
  exposure = "A",
  mediator = "M_star",
  outcome = "Y",
  confounders = "C1",
  misclassified_variable = "mediator",
  sensitivity_region = sens_region,
  n_grid = 10,
  ci_method = "analytic", ci_n_boot = 50
)

# View results
print(bounds)
summary(bounds)
bounds@analytic_ci$NDE   # raw [L, U] plus Imbens-Manski confidence interval

# Visualize
sensitivity_plot(bounds, param = "psi_sn")
```

`check_compatibility` *Check Compatibility of Misclassification Parameters*

Description

Tests whether a specific set of misclassification parameters is compatible with the observed data by checking testable implications. Returns detailed diagnostics about which constraints are satisfied or violated.

Usage

```
check_compatibility(
  data,
  exposure,
```

```
  mediator,
  outcome,
  confounders = NULL,
  psi,
  misclassified_variable = c("exposure", "mediator"),
  return_details = TRUE,
  tolerance = 1e-06
)
```

Arguments

`data`	Data frame containing the observed variables
`exposure`	Character string. Name of exposure variable
`mediator`	Character string. Name of mediator variable
`outcome`	Character string. Name of outcome variable
`confounders`	Character vector. Names of confounding variables
`psi`	Named list containing misclassification parameters: • `sn0`: Baseline sensitivity • `sp0`: Baseline specificity • `psi_sn`: Differential sensitivity (odds ratio) • `psi_sp`: Differential specificity (odds ratio)
`misclassified_variable`	Character string. Either "exposure" or "mediator"
`return_details`	Logical. If TRUE, returns detailed stratum-level diagnostics. Default is TRUE.
`tolerance`	Numeric. Tolerance for numerical precision when checking constraints. Default is 1e-6. On the mediator path, bound_ne applies the same constraints with no tolerance, so this function is the more lenient of the two there: a `psi` whose implied probabilities fall outside [0, 1] by less than `tolerance` is reported compatible here but rejected by `bound_ne()`.

Details

This function implements the testable implications derived in Propositions 4.1 and 5.1 of the paper. For mediator misclassification, it checks whether the observed data can be explained by any true causal parameters given the specified misclassification mechanism. For exposure misclassification, it checks the likelihood ratio constraints on observed probabilities.

The function is useful for:

- Testing specific hypotheses about misclassification parameters
- Understanding which constraints are most informative
- Debugging sensitivity analyses
- Generating diagnostic plots

Value

A list with class `compatibility_test` containing:

`compatible` Logical. TRUE if parameters are compatible

`psi` The tested parameter set

`n_constraints_total` Total number of testable constraints

`n_constraints_satisfied` Number of satisfied constraints

`n_constraints_violated` Number of violated constraints

`violated_constraints` Data frame of violated constraints (if any)

`implied_probabilities` Solved true probabilities (if compatible)

`stratum_details` Stratum-level diagnostics (if return_details=TRUE)

See Also

`bound_ne`, `falsification_summary`

Examples

```
data("heals_data")

# Test non-differential misclassification with high accuracy
test_ndm <- check_compatibility(
  data = heals_data,
  exposure = "A_star",
  mediator = "M",
  outcome = "Y",
  confounders = c("age", "male", "smoking", "bmi"),
  psi = list(sn0 = 0.90, sp0 = 0.90, psi_sn = 1.0, psi_sp = 1.0),
  misclassified_variable = "exposure"
)

print(test_ndm)

# Test strong differential misclassification
test_strong_dm <- check_compatibility(
  data = heals_data,
  exposure = "A_star",
  mediator = "M",
  outcome = "Y",
  confounders = c("age", "male", "smoking", "bmi"),
  psi = list(sn0 = 0.85, sp0 = 0.85, psi_sn = 3.0, psi_sp = 1.0),
  misclassified_variable = "exposure"
)
```

```
print(test_strong_dm)
```

---

compare_bounds *Compare Bounds Across Multiple Analyses*

Description

Compare partial identification bounds from multiple analyses (e.g., different sensitivity regions, different datasets, different assumptions).

Usage

```
compare_bounds(bounds_list, labels = NULL)
```

Arguments

bounds_list	A named list of medrobust_bounds objects.
labels	Optional character vector of labels for each analysis. If NULL, uses names from bounds_list.

Value

A data frame comparing the bounds, and optionally a plot.

Examples

```
# Compare bounds under different sensitivity assumptions
sim <- simulate_dm_data(
  n = 500,
  true_params = list(beta_AM = log(2.5), theta_AY = log(1.5), theta_MY = log(2.5)),
  dm_params = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
  misclass_type = "mediator", confounders = 1, seed = 1
)
args0 <- list(
  data = sim@observed, exposure = "A", mediator = "M_star", outcome = "Y",
  confounders = "C1", misclassified_variable = "mediator", n_grid = 10
)
set.seed(1)
bounds1 <- do.call(bound_ne, c(args0, list(sensitivity_region = list(
  sn0_range = c(0.82, 0.97), sp0_range = c(0.82, 0.97),
  psi_sn_range = c(0.85, 1.3), psi_sp_range = c(0.85, 1.3)))))
bounds2 <- do.call(bound_ne, c(args0, list(sensitivity_region = list(
  sn0_range = c(0.80, 0.99), sp0_range = c(0.80, 0.99),
  psi_sn_range = c(0.8, 1.5), psi_sp_range = c(0.8, 1.5)))))

comparison <- compare_bounds(
  list(conservative = bounds1, liberal = bounds2)
```

```
)
print(comparison)
```

`compatibility_test` *Compatibility Test Class*

Description

S7 class for storing results of compatibility tests for specific misclassification parameter values.

Arguments

`compatible`	Logical indicating if parameters are compatible with data
`psi`	List of misclassification parameters tested
`sn1`	Sensitivity when Y=1 (implied from psi)
`sp1`	Specificity when Y=1 (implied from psi)
`n_constraints_total`	Total number of testable constraints
`n_constraints_satisfied`	Number of constraints satisfied
`n_constraints_violated`	Number of constraints violated
`violated_constraints`	Data frame with details of violated constraints
`implied_probabilities`	List of implied probability distributions, or `NULL` when the data are incompatible with `psi`
`stratum_details`	List with stratum-specific details, or `NULL` when they were not requested or the test exited before any stratum was checked
`misclassified_variable`	Character: "exposure" or "mediator"
`reason`	Character describing reason for incompatibility (if any)

Value

An S7 object of class `compatibility_test` holding the outcome of the data-compatibility (falsification) test, including the satisfied and violated testable constraints.

compute_bound_se *Compute Standard Errors for Bounds*

Description

Estimate standard errors for the lower and upper bounds using the bootstrap distribution.

Usage

```
compute_bound_se(bootstrap_results)
```

Arguments

bootstrap_results	The bootstrap_results property of a bound_ne fit with bootstrap = TRUE, or the list returned by compute_bootstrap_ci()

Value

Named vector of standard errors

extract_bounds *Extract Compatible Parameter Sets*

Description

Extract the compatible parameter sets from a bounds analysis.

Usage

```
extract_bounds(bounds_object)
```

Arguments

bounds_object	An object of class medrobust_bounds

Value

A data frame containing compatible parameter sets

Examples

```
sim <- simulate_dm_data(
  n = 2000,
  true_params = list(beta_AM = log(2.5), theta_AY = log(1.5), theta_MY = log(2.5)),
  dm_params = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
  misclass_type = "mediator", confounders = 1, seed = 1
)
bounds <- bound_ne(
  data = sim@observed, exposure = "A", mediator = "M_star", outcome = "Y",
  confounders = "C1", misclassified_variable = "mediator",
  sensitivity_region = list(
    sn0_range = c(0.80, 0.99), sp0_range = c(0.80, 0.99),
    psi_sn_range = c(0.8, 1.5), psi_sp_range = c(0.8, 1.5)
  ),
  n_grid = 10, verbose = FALSE
)
head(extract_bounds(bounds))
```

---

extract_falsified_region

Extract Falsified Region

Description

Extract the parameter sets that the grid search evaluated and the data falsified. Works for every `grid_method`, because bound_ne records each evaluated set with its verdict.

Usage

```
extract_falsified_region(bounds_object)
```

Arguments

`bounds_object`	An object of class `medrobust_bounds`

Value

A data frame with columns `sn0`, `sp0`, `psi_sn` and `psi_sp`, one row per falsified parameter set.

`falsification_summary` *Summarize Falsification Results*

Description

Provides a detailed summary of which regions of the sensitivity parameter space are empirically falsified by the data. Helps understand which assumptions about misclassification are most constrained by the observed data.

Usage

```
falsification_summary(
  bounds_object,
  by_parameter = TRUE,
  n_bins = 10,
  plot = TRUE
)
```

Arguments

`bounds_object`	An object of class `medrobust_bounds` returned by `bound_ne`
`by_parameter`	Logical. If TRUE, breaks down falsification by each parameter dimension. Default is TRUE.
`n_bins`	Integer. Number of bins for discretizing parameters when computing falsification rates. Default is 10.
`plot`	Logical. If TRUE, generates visualization of falsification patterns. Default is TRUE.

Details

This function uses every parameter set `bound_ne()` evaluated (its `evaluated_sets` property) to show which regions of the sensitivity space are ruled out by the testable implications. The falsification rate of a bin is the share of the sets evaluated in that bin that the data falsified; a bin in which no set was evaluated has rate `NA`. High falsification rates indicate that the data are informative about that particular parameter.

The falsification analysis is useful for:

- Understanding which misclassification assumptions are most constrained
- Identifying whether bounds are wide due to weak data vs. wide sensitivity region
- Guiding choice of sensitivity parameters for future studies
- Assessing whether additional data would meaningfully narrow bounds

Value

A list of class `falsification_summary` containing:

`overall`	Overall falsification rate
`by_parameter`	Parameter-specific falsification rates (if by_parameter=TRUE)
`joint_falsification`	2D falsification patterns
`most_constrained`	Parameters that are most falsified
`least_constrained`	Parameters that are least falsified
`plot`	ggplot2 object (if plot=TRUE)

See Also

`bound_ne`, `check_compatibility`

Examples

```
# Compute bounds first (see ?bound_ne)
sim <- simulate_dm_data(
  n = 8000,
  true_params = list(beta_AM = log(2.5), theta_AY = log(1.5), theta_MY = log(2.5)),
  dm_params = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
  misclass_type = "mediator", confounders = 1, seed = 1
)
set.seed(1)
bounds <- bound_ne(
  data = sim@observed, exposure = "A", mediator = "M_star", outcome = "Y",
  confounders = "C1", misclassified_variable = "mediator",
  sensitivity_region = list(
    sn0_range = c(0.80, 0.99), sp0_range = c(0.80, 0.99),
    psi_sn_range = c(0.8, 1.5), psi_sp_range = c(0.8, 1.5)
  ),
  n_grid = 10
)

falsif <- falsification_summary(bounds)
print(falsif)

# View falsification plot
falsif@plot
```

format_effect *Format Effect Estimate for Reporting*

Description

Format an effect estimate as a string for tables/reports.

Usage

```
format_effect(estimate, effect_scale = "OR", digits = 2, ci = NULL)
```

Arguments

estimate	Numeric effect estimate (or bounds)
effect_scale	Character string: "OR", "RR", or "RD"
digits	Integer: number of decimal places
ci	Optional: confidence interval (length 2 vector)

Value

Character string

Examples

```
format_effect(1.48)
format_effect(1.48, ci = c(1.21, 1.81))   # point estimate with a CI
format_effect(c(1.12, 1.37))              # bounds [L, U]
format_effect(0.031, effect_scale = "RD", digits = 3)
```

gesthtn *Gestational Hypertension as a Differentially Misclassified Binary Mediator*

Description

A random sample of U.S. live births used to illustrate partial-identification bounds when a binary mediator is differentially misclassified, without validation data. The mediator—gestational hypertension as recorded on the birth certificate—is a textbook differentially misclassified variable: its birth-certificate sensitivity is poor and reporting accuracy plausibly depends on the outcome (preterm birth). This is the substantive triad analyzed by the closest prior point-identification method (Hochstedler Webb and Wells, 2025).

Usage

```
gesthtn
```

Format

A data frame with 5,000 observations and 4 binary variables:

A Binary. Advanced maternal age (1 = age >= 35 years, 0 = age < 35).

M_star Binary. Recorded (surrogate) gestational hypertension from the birth certificate (1 = Yes, 0 = No). A misclassified measurement of the true mediator; the true status is not observed.

Y Binary. Preterm birth (1 = gestational age < 37 weeks, 0 = term).

C1 Binary. Parity (1 = one or more prior live births, 0 = none).

Details

Source data A fixed-seed (20260614) random sample of 5,000 from 3,662,426 complete cases in the 2021 U.S. National Center for Health Statistics (NCHS) Natality (birth-certificate) file, public-domain U.S.-government data obtained via the NBER mirror. Variables were derived as above from `mager`, `rf_ghype`, `combgest`, and `priorlive`; see `system.file("scripts", "prepare_gesthtn.R", package = "medrobust")`.

Misclassification Validation of the 2003 revised birth certificate against the medical record (Dietz et al., 2015) found the gestational-hypertension item to have *poor* sensitivity (< 70%) and *excellent* specificity (> 90%); differential (outcome-dependent) sensitivity is plausible. These motivate a sensitivity region with baseline sensitivity in roughly [0.50, 0.70], specificity in [0.90, 0.99], and a differential-sensitivity odds ratio at or above 1.

Source

NCHS Natality 2021 (U.S. birth certificates), via the NBER mirror https://data.nber.org/nvss/natality/csv/2021/natality2021us.csv. Misclassification context: Dietz et al. (2015), *Public Health Reports* 130(1):60-70, doi:10.1177/003335491513000108. Prior art on this application: Hochstedler Webb and Wells (2025), *Statistical Methods in Medical Research* 34(5):1037-1059, doi:10.1177/09622802251316970.

Examples

```
data("gesthtn")
table(gesthtn$M_star, gesthtn$Y)

# Partial-identification bounds for the (differentially misclassified) mediator;
# see ?bound_ne and the article
# https://data-wise.github.io/medrobust/articles/gesthtn-bounds.html
region <- sensitivity_region(
  sn0_range = c(0.50, 0.70), sp0_range = c(0.90, 0.99),
  psi_sn_range = c(1.0, 3.0), psi_sp_range = c(1.0, 1.0)
)
```

heals_data *Synthetic HEALS Data with Differential Measurement Error*

Description

A synthetic dataset based on the Health Effects of Arsenic Longitudinal Study (HEALS) that demonstrates differential measurement error in exposure assessment. The dataset includes both observed (misclassified) and true (ground truth) arsenic exposure, allowing for validation of sensitivity analysis methods.

Usage

```
heals_data
```

Format

A data frame with 450 observations and 9 variables:

id Integer. Unique subject identifier (1 to 450).

Y Binary. Cardiovascular disease status (0 = Absent, 1 = Present).

M Binary. Elevated inflammation marker (sVCAM-1) (0 = Low, 1 = High).

A_star Binary. Self-reported arsenic exposure (0 = No, 1 = Yes). Subject to severe differential recall bias.

A_true Binary. True arsenic exposure status (0 = No, 1 = Yes). Included for validation; not available in real studies.

age Numeric. Age in years (range: 18-75, mean ~42.8).

male Binary. Sex (0 = Female, 1 = Male).

smoking Binary. Ever smoker status (0 = Never, 1 = Ever).

bmi Numeric. Body mass index in kg/m² (mean ~19.7).

Details

Data Generation The data were generated to mimic the HEALS cohort using published effect sizes and demographic profiles from:

- Ahsan et al. (2006) - Baseline demographics
- Chen et al. (2007) - Arsenic-inflammation relationship
- Argos et al. (2010) - Arsenic-CVD relationship (HR ~ 1.92)

Measurement Error Model Differential misclassification was introduced with:

- Cases (Y=1): Sensitivity = 0.90, Specificity = 0.55 (over-reporting)
- Controls (Y=0): Sensitivity = 0.60, Specificity = 0.90

This creates severe recall bias where CVD cases over-report arsenic exposure, inflating the naive effect estimate by approximately 70

True vs. Naive Effects

- True Natural Direct Effect: OR ~ 1.68
- Naive NDE (using A_star): OR ~ 2.85
- Bias amplification: ~70

Source

Synthetic data generated using methods described in the article "Synthetic HEALS Data: Ground Truth with Differential Measurement Error" on the package website: https://data-wise.github.io/medrobust/articles/heals-synthetic-data.html.

Examples

```
# Load the data
data("heals_data")
str(heals_data)

# Examine prevalence
table(heals_data$A_star, heals_data$Y)

# Compare naive vs. true effect
naive_mod <- glm(Y ~ A_star + M + age + male + smoking + bmi,
                 data = heals_data, family = binomial)
true_mod <- glm(Y ~ A_true + M + age + male + smoking + bmi,
                data = heals_data, family = binomial)

cat("Naive NDE:", round(exp(coef(naive_mod)["A_star"]), 2), "\n")
cat("True NDE:", round(exp(coef(true_mod)["A_true"]), 2), "\n")

# To compute partial-identification bounds for the (potentially
# misclassified) exposure A_star, pass this data to bound_ne();
# see ?bound_ne for a complete, runnable example.
```

---

medrobust_bounds *Medrobust Bounds Class*

Description

S7 class for storing partial identification bounds for natural direct and indirect effects under differential misclassification.

Arguments

`NIE_lower`	Lower bound for Natural Indirect Effect
`NIE_upper`	Upper bound for Natural Indirect Effect
`NDE_lower`	Lower bound for Natural Direct Effect
`NDE_upper`	Upper bound for Natural Direct Effect
`compatible_sets`	Data frame of parameter sets compatible with data
`n_compatible`	Number of compatible parameter sets
`n_evaluated`	Total number of parameter sets evaluated
`falsified_proportion`	Proportion of parameter space falsified
`effect_scale`	Character: "OR", "RR", or "RD"
`misclassified_variable`	Character: "exposure" or "mediator"
`sensitivity_region`	Sensitivity region specification
`naive_estimates`	List of naive effect estimates
`bootstrap_results`	Bootstrap inference results (if computed)
`data_summary`	Summary statistics from the data
`evaluated_sets`	Data frame of every parameter set the grid search evaluated (`sn0`, `sp0`, `psi_sn`, `psi_sp`) with a logical `compatible` column; `NULL` when no search ran
`call`	The function call that created this object

Value

An S7 object of class `medrobust_bounds` containing the partial identification bounds for the natural direct and indirect effects, the parameter sets compatible with the data, and the associated inference and diagnostic summaries.

`new_falsification_summary` *Create Falsification Summary Object*

Description

Low-level constructor for falsification_summary S7 objects. Most users should use the falsification_summary() function which analyzes bounds.

Usage

```
new_falsification_summary(
  overall,
  n_evaluated,
  n_compatible,
  n_falsified,
  by_parameter = NULL,
  joint_falsification = NULL,
  most_constrained = character(0),
  least_constrained = character(0),
  plot = NULL
)
```

Arguments

`overall`	Overall falsification rate
`n_evaluated`	Number of parameter sets evaluated
`n_compatible`	Number of compatible parameter sets
`n_falsified`	Number of falsified parameter sets
`by_parameter`	Parameter-specific falsification (optional)
`joint_falsification`	Joint falsification patterns (optional)
`most_constrained`	Most constrained parameters (optional)
`least_constrained`	Least constrained parameters (optional)
`plot`	ggplot2 object (optional)

Value

A falsification_summary S7 object

Examples

```
new_falsification_summary(
  overall = 0.25, n_evaluated = 100L, n_compatible = 75L, n_falsified = 25L
)
```

nhanes_pa *Physical Inactivity as a Differentially Misclassified Binary Exposure*

Description

A pooled sample of U.S. adults from NHANES used to illustrate partial-identification bounds when a binary *exposure* is differentially misclassified, without validation data. The exposure—self-reported physical inactivity—is a textbook differentially misclassified variable: it has no biomarker, relies on participant recall and report, and reporting accuracy plausibly depends on health status (and thus on the outcome). The mediator, laboratory-measured systemic inflammation (hs-CRP), is by contrast measured *without error*. `nhanes_pa` is the exposure-side mirror image of `gesthtn` (which misclassifies the *mediator*), completing the mediator/exposure symmetry of the package's worked examples.

Usage

```
nhanes_pa
```

Format

A data frame with approximately 9,906 observations and 6 binary variables (every column is an `integer` in {0, 1}):

A_star Binary. Self-reported physical inactivity (1 = inactive, 0 = active). A misclassified (surrogate) measurement of the true exposure; "inactive" means leisure-time aerobic activity below the 2018 Physical Activity Guidelines threshold of 150 moderate-equivalent minutes per week (vigorous minutes counted double). The true activity status is not observed.

M Binary. Elevated systemic inflammation, hs-CRP >= 3 mg/L (1 = elevated, 0 = not). Laboratory-measured, treated as error-free.

Y Binary. Prevalent cardiovascular disease (1 = any of coronary heart disease, angina, myocardial infarction, or stroke; 0 = none).

C1 Binary. Older age (1 = age >= 50 years, 0 = age < 50).

C2 Binary. Female sex (1 = female, 0 = male).

C3 Binary. Obesity (1 = BMI >= 30, 0 = BMI < 30).

Details

Source data Pooled NHANES 2015–2016 (`_I`) and 2017–2018 (`_J`) cycles, adults aged 20 and older, restricted to complete cases on the six modeled binaries. Unlike `gesthtn` (a fixed-seed random sample drawn from millions of births), `nhanes_pa` ships the *full* complete-case sample with **no random sampling**: the data are tiny (six binary columns), and the complete-case construction is deterministic, so the dataset is reproducible by construction with no seed. The eight `C1 x C2 x C3` strata are well-populated (each roughly 843–1,588), which keeps `bound_ne(confounders = c("C1", "C2", "C3"))` feasible across the differential-sensitivity sweep. Variables were derived from the PAQ (physical-activity questionnaire), HSCRP (high-sensitivity C-reactive protein), MCQ (medical

conditions), DEMO (demographics), and BMX (body measures) files; see `system.file("scripts", "prepare_nhanes_pa.R", package = "medrobust")`.

Misclassification The misclassified variable here is a **self-reported exposure** in a *cross-sectional* survey, so the relevant mechanism is differential *reporting*: sensitivity and specificity for self-reported inactivity may depend on the prevalent outcome (e.g., respondents with cardiovascular disease may report activity differently from healthy respondents). This is outcome-dependent reporting error, **not** case-control recall bias—there is no retrospective recall of past exposure conditional on diagnosis. With no validation data, an analyst can posit plausible accuracy (e.g., baseline sensitivity and specificity each in [0.80, 0.95]) and sweep a differential-sensitivity odds ratio (ψ_{Sn}) at or above 1 to probe robustness.

Source

CDC National Health and Nutrition Examination Survey (NHANES), public-domain U.S.-government data for cycles 2015–2016 and 2017–2018, obtained via the **nhanesA** package (https://cran.r-project.org/package=nhanesA). Survey home: https://www.cdc.gov/nchs/nhanes/. Activity threshold: 2018 Physical Activity Guidelines for Americans (150 moderate-equivalent min/week).

Examples

```
data("nhanes_pa")
table(A_star = nhanes_pa$A_star, Y = nhanes_pa$Y)

# Partial-identification bounds for the (differentially misclassified) exposure;
# see ?bound_ne and the article
# https://data-wise.github.io/medrobust/articles/nhanes_pa-bounds.html
region <- sensitivity_region(
  sn0_range = c(0.80, 0.95), sp0_range = c(0.80, 0.95),
  psi_sn_range = c(1.0, 2.0), psi_sp_range = c(1.0, 1.0)
)
```

---

plot_bootstrap_distribution

Plot Bootstrap Distribution

Description

Visualize the bootstrap distribution of bounds.

Usage

```
plot_bootstrap_distribution(
  bootstrap_results,
  effect = "NIE",
  original_bounds = NULL
)
```

Arguments

bootstrap_results	The bootstrap_results property of a bound_ne fit with bootstrap = TRUE, or the list returned by compute_bootstrap_ci()
effect	Character string: "NIE" or "NDE"
original_bounds	Original bound estimates (optional)

Value

ggplot2 object

---

power_analysis *Power Analysis for Partial Identification Bounds*

Description

Conducts simulation-based power analysis to determine the sample size needed to achieve a target bound width or to rule out the null hypothesis with high probability.

Usage

```
power_analysis(
  true_params,
  dm_params,
  sensitivity_region,
  misclass_type = c("exposure", "mediator"),
  sample_sizes = seq(100, 1000, by = 100),
  target_width = NULL,
  target_power = 0.8,
  alpha = 0.05,
  effect = c("NIE", "NDE"),
  n_sim = 100,
  n_grid = 30,
  confounders = 1,
  parallel = TRUE,
  n_cores = NULL,
  verbose = TRUE,
  seed = 12345
)
```

Arguments

true_params	Named list of true causal parameters (see simulate_dm_data)
dm_params	Named list of misclassification parameters

`sensitivity_region`
: Named list defining Theta_Psi for bound_ne

`misclass_type`
: Character string. "exposure" or "mediator"

`sample_sizes`
: Integer vector. Sample sizes to evaluate. Default is seq(100, 1000, by = 100).

`target_width`
: Numeric. Target bound width. If specified, finds minimum sample size to achieve this width with high probability. Default is NULL.

`target_power`
: Numeric. Target power for rejecting null. Default is 0.80.

`alpha`
: Numeric. Significance level. Default is 0.05.

`effect`
: Character string. Which effect to power for: "NIE" or "NDE". Default is "NIE".

`n_sim`
: Integer. Number of simulation replicates per sample size. Default is 100.

`n_grid`
: Integer. Grid resolution for bound_ne. Default is 30.

`confounders`
: Integer. Number of confounders. Default is 1.

`parallel`
: Logical. Use parallel processing? Default is TRUE.

`n_cores`
: Integer. Number of cores. Default is NULL (auto-detect).

`verbose`
: Logical. Print progress? Default is TRUE.

`seed`
: Integer. Random seed. Default is 12345.

Value

An S7 object of class `power_analysis_result` containing:

`power_curve`
: Data frame with power by sample size

`true_effect`
: True effect value

`target_power`
: Target power level

`target_width`
: Target bound width (if specified)

`recommended_n_power`
: Recommended sample size for target power

`recommended_n_width`
: Recommended sample size for target width

`simulation_params`
: Parameters used for simulation

See Also

`simulate_dm_data`, `bound_ne`

Examples

```
# A power analysis runs many bootstrap-bound replications across several
# sample sizes, so it is computationally intensive; it is wrapped in
# \dontrun{} to keep checks fast. Reduce sample_sizes / n_sim to run quickly.
## Not run:
# Power analysis for exposure DM with moderate effects
power_result <- power_analysis(
  true_params = list(beta_AM = 0.405, theta_AY = 0.405, theta_MY = 0.405),
```

```
  dm_params = list(sn0 = 0.85, sp0 = 0.85, psi_sn = 1.5, psi_sp = 1.0),
  sensitivity_region = list(
    sn0_range = c(0.80, 0.90),
    sp0_range = c(0.80, 0.90),
    psi_sn_range = c(1.0, 2.0),
    psi_sp_range = c(1.0, 1.0)
  ),
  misclass_type = "exposure",
  sample_sizes = c(200, 400, 600, 800, 1000),
  target_width = 0.3,
  target_power = 0.80,
  n_sim = 100
)

print(power_result)
plot(power_result)

## End(Not run)
```

`power_analysis_result` *Power Analysis Result Class*

Description

S7 class for storing power analysis results for partial identification bounds.

Arguments

`power_curve`	Data frame with power and width by sample size
`true_effect`	True effect value used in simulations
`target_power`	Target power level for sample size recommendations
`target_width`	Target bound width for sample size recommendations
`recommended_n_power`	Recommended sample size to achieve target power
`recommended_n_width`	Recommended sample size to achieve target width
`simulation_params`	List of simulation parameters used

Value

An S7 object of class `power_analysis_result` holding the power and bound-width curves across sample sizes and the recommended sample sizes for the target power and width.

print.compatibility_test *Print Method for compatibility_test*

Description

Prints a formatted report of a falsification test returned by `check_compatibility`. This is an S7 method for the base generic; call it as `print(x)`.

Arguments

`x`	An object of class `compatibility_test`.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object.

See Also

`check_compatibility`, `summary.compatibility_test`

sensitivity_plot *Create Sensitivity Analysis Plots*

Description

Generate publication-quality visualizations of partial identification bounds as a function of sensitivity parameters.

Usage

```
sensitivity_plot(
  bounds_object,
  param = "psi_sn",
  effect = c("both", "NIE", "NDE"),
  plot_type = c("bounds", "heatmap", "contour"),
  show_naive = TRUE,
  show_null = TRUE,
  color_scheme = "default",
  theme = "bw",
  ...
)
```

Arguments

bounds_object	An object of class medrobust_bounds returned by bound_ne.
param	Character vector specifying which parameter(s) to plot on the x-axis. Options: "psi_sn", "psi_sp", "sn0", "sp0". Can specify multiple for faceted plots.
effect	Character string: "NIE", "NDE", or "both" (default).
plot_type	Character string specifying plot type: • "bounds" (default): Line plot showing upper and lower bounds • "heatmap": 2D heatmap of bound width (requires two parameters) • "contour": Contour plot of bounds (requires two parameters)
show_naive	Logical indicating whether to overlay the naive estimate (assuming no misclassification). Default is TRUE.
show_null	Logical indicating whether to show horizontal line at null value (effect = 1 for OR/RR, effect = 0 for RD). Default is TRUE.
color_scheme	Character string specifying color palette: "default", "viridis", "colorblind", "grayscale".
theme	Character string for ggplot2 theme: "bw" (default), "minimal", "classic", "void".
...	Additional arguments passed to ggplot2 functions.

Details

The function creates different plot types to visualize sensitivity analysis:

Bounds plot: Shows how the lower and upper bounds vary as a function of one sensitivity parameter, with other parameters held fixed or averaged.

Heatmap: Shows the width of the identification region (upper - lower) as a function of two sensitivity parameters simultaneously.

Contour plot: Shows contour lines of the bounds in 2D parameter space.

Value

A ggplot2 object that can be further customized or saved.

Examples

```
# Compute bounds to plot (see ?bound_ne)
sim <- simulate_dm_data(
  n = 8000,
  true_params = list(beta_AM = log(2.5), theta_AY = log(1.5), theta_MY = log(2.5)),
  dm_params = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
  misclass_type = "mediator", confounders = 1, seed = 1
)
set.seed(1)
bounds <- bound_ne(
  data = sim@observed, exposure = "A", mediator = "M_star", outcome = "Y",
  confounders = "C1", misclassified_variable = "mediator",
  sensitivity_region = list(
    sn0_range = c(0.80, 0.99), sp0_range = c(0.80, 0.99),
    psi_sn_range = c(0.8, 1.5), psi_sp_range = c(0.8, 1.5)
```

```
  ),
  n_grid = 10
)

# Basic bounds plot
sensitivity_plot(bounds, param = "psi_sn", effect = "NIE")

# Show both effects
sensitivity_plot(bounds, param = "psi_sn", effect = "both")

# Customize the returned ggplot object
p <- sensitivity_plot(bounds, param = "psi_sn") +
  ggplot2::labs(title = "My Custom Title") +
  ggplot2::theme(legend.position = "bottom")
p
```

`sensitivity_region` *Create Sensitivity Region*

Description

Constructor for sensitivity_region S7 objects. Issues a warning if the region may be non-informative (Sn + Sp <= 1).

Usage

```
sensitivity_region(sn0_range, sp0_range, psi_sn_range, psi_sp_range)
```

Arguments

`sn0_range`	Numeric vector of length 2: [min, max] for baseline sensitivity
`sp0_range`	Numeric vector of length 2: [min, max] for baseline specificity
`psi_sn_range`	Numeric vector of length 2: [min, max] for sensitivity OR
`psi_sp_range`	Numeric vector of length 2: [min, max] for specificity OR

Value

A sensitivity_region S7 object

Examples

```
region <- sensitivity_region(
  sn0_range = c(0.80, 0.95), sp0_range = c(0.85, 0.99),
  psi_sn_range = c(1, 2), psi_sp_range = c(1, 1)
)
region
```

simulate_dm_data *Simulate Data with Differential Misclassification*

Description

Generates synthetic data with known differential misclassification for power analysis, methods validation, and simulation studies. Allows control over true causal parameters and misclassification mechanisms.

Usage

```
simulate_dm_data(
  n = 500,
  true_params = list(beta_AM = 0.405, theta_AY = 0.405, theta_MY = 0.405),
  dm_params = list(sn0 = 0.85, sp0 = 0.85, psi_sn = 1.5, psi_sp = 1),
  misclass_type = c("exposure", "mediator"),
  confounders = 1,
  confounder_params = list(type = "binary", effect_on_A = 0.3, effect_on_M = 0.3,
    effect_on_Y = 0.3),
  effect_modification = FALSE,
  interaction_coef = 0,
  seed = NULL,
  return_truth = TRUE,
  return_params = TRUE
)
```

Arguments

n	Integer. Sample size. Default is 500.
true_params	Named list of true causal parameters: • beta_AM: Effect of A on M (log-odds scale) • theta_AY: Direct effect of A on Y (log-odds scale) • theta_MY: Effect of M on Y (log-odds scale) • baseline_M: Baseline probability of M when A=0 (optional) • baseline_Y: Baseline probability of Y when A=0, M=0 (optional)
dm_params	Named list of misclassification parameters: • sn0: Baseline sensitivity (Y=0) • sp0: Baseline specificity (Y=0) • psi_sn: Differential sensitivity (odds ratio) • psi_sp: Differential specificity (odds ratio)
misclass_type	Character string. Either "exposure" or "mediator" to indicate which variable is misclassified. Default is "exposure".
confounders	Integer. Number of confounding variables to include. Default is 1. Set to 0 for no confounders.

`confounder_params`
: Named list controlling confounder generation:
  - `type`: "binary" or "continuous". Default is "binary".
  - `effect_on_A`: Effect size on exposure (log-odds). Default is 0.3.
  - `effect_on_M`: Effect size on mediator (log-odds). Default is 0.3.
  - `effect_on_Y`: Effect size on outcome (log-odds). Default is 0.3.

`effect_modification`
: Logical. Should there be effect modification (interaction between A and M on Y)? Default is FALSE.

`interaction_coef`
: Numeric. Interaction coefficient if effect_modification=TRUE. Default is 0.

`seed`
: Integer. Random seed for reproducibility. If NULL, no seed is set.

`return_truth`
: Logical. Should true (unobserved) values be returned? Default is TRUE.

`return_params`
: Logical. Should true causal effects be calculated and returned? Default is TRUE.

Value

An S7 object of class `simulated_dm_data` containing:

`observed`
: Data frame with observed (potentially misclassified) variables

`truth`
: Data frame with true (unobserved) values (if return_truth=TRUE)

`true_effects`
: List of true causal effects (if return_params=TRUE)

`generation_params`
: Parameters used for data generation

`misclassification_applied`
: Summary of applied misclassification

See Also

`bound_ne`, `power_analysis`

Examples

```
# Basic simulation with moderate effects and mild DM
sim_data <- simulate_dm_data(
  n = 500,
  true_params = list(
    beta_AM = 0.405,   # OR = 1.5
    theta_AY = 0.405,  # OR = 1.5
    theta_MY = 0.405   # OR = 1.5
  ),
  dm_params = list(
    sn0 = 0.85,
    sp0 = 0.85,
    psi_sn = 1.5,
    psi_sp = 1.0
  ),
```

```
  misclass_type = "exposure",
  seed = 12345
)

# Use in analysis
set.seed(1)
bounds <- bound_ne(
  data = sim_data@observed,
  exposure = "A_star",
  mediator = "M",
  outcome = "Y",
  confounders = "C1",
  misclassified_variable = "exposure",
  sensitivity_region = list(
    sn0_range = c(0.80, 0.90),
    sp0_range = c(0.80, 0.90),
    psi_sn_range = c(1.0, 2.0),
    psi_sp_range = c(1.0, 1.0)
  ),
  n_grid = 10
)
```

`simulated_dm_data` *Simulated Data with Differential Misclassification Class*

Description

S7 class for storing simulated data with known differential misclassification, used for power analysis and methods validation.

Arguments

`observed`	Data frame with observed (potentially misclassified) variables
`truth`	Data frame with true (unobserved) values
`true_effects`	List of true causal effects
`generation_params`	List of parameters used to generate the data
`misclassification_applied`	List of misclassification parameters applied

Value

An S7 object of class `simulated_dm_data` holding the simulated observed data together with the true (unobserved) values, the true causal effects, and the data-generating parameters.

summary.compatibility_test

Summary Method for compatibility_test

Description

Prints a detailed summary of a falsification test returned by `check_compatibility`. This is an S7 method for the base generic; call it as `summary(object)`.

Arguments

`object`	An object of class `compatibility_test`.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object.

See Also

`check_compatibility`, `print.compatibility_test`

test_multiple_hypotheses

Test Multiple Hypotheses

Description

Test multiple hypotheses about misclassification parameters simultaneously.

Usage

```
test_multiple_hypotheses(
  data,
  exposure,
  mediator,
  outcome,
  confounders,
  psi_list,
  misclassified_variable
)
```

Arguments

data	Data frame
exposure	Character string
mediator	Character string
outcome	Character string
confounders	Character vector
psi_list	List of parameter sets to test, each a list with sn0, sp0, psi_sn and psi_sp. Names label the rows of the result; unnamed entries are labeled H1, H2, ... by position.
misclassified_variable	Character string

Value

Data frame with test results for each hypothesis

Examples

```
sim <- simulate_dm_data(
  n = 2000,
  true_params = list(beta_AM = log(2.5), theta_AY = log(1.5), theta_MY = log(2.5)),
  dm_params = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
  misclass_type = "mediator", confounders = 1, seed = 1
)

# psi_list must be named: the names label the rows of the result
test_multiple_hypotheses(
  data = sim@observed, exposure = "A", mediator = "M_star", outcome = "Y",
  confounders = "C1", misclassified_variable = "mediator",
  psi_list = list(
    accurate = list(sn0 = 0.9, sp0 = 0.9, psi_sn = 1, psi_sp = 1),
    poor     = list(sn0 = 0.6, sp0 = 0.6, psi_sn = 1, psi_sp = 1)
  )
)
```

Index

∗ **datasets**
    gesthtn, 21
    heals_data, 23
    nhanes_pa, 27

as.data.frame.medrobust_bounds, 3
as.list.sensitivity_region, 3
as_sensitivity_region, 4

bootstrap_results, 5
bootstrap_width_summary, 6
bound_ci, 6, *9*
bound_ne, *3, 4*, *6, 7*, 8, *13, 14*, *17–20*, *29, 30*, *33*, *36*

check_compatibility, *11*, 12, *20*, *32*, *38*
compare_bounds, 15
compatibility_test, 16, *32*, *38*
compute_bound_se, 17

extract_bounds, 17
extract_falsified_region, *11*, 18

falsification_summary, *11*, *14*, 19
format_effect, 21

gesthtn, 21, *27*

heals_data, 23

medrobust_bounds, *3*, 24

new_falsification_summary, 25
nhanes_pa, 27

plot_bootstrap_distribution, 28
power_analysis, 29, *36*
power_analysis_result, 31
print.compatibility_test, 32, *38*

sensitivity_plot, *11*, 32
sensitivity_region, *3, 4*, 34
simulate_dm_data, *29, 30*, 35
simulated_dm_data, 37
summary.compatibility_test, *32*, 38

test_multiple_hypotheses, 38