Package: causaldef (via r-universe)

May 31, 2026

Type Package

Title Decision-Theoretic Causal Diagnostics via Le Cam Deficiency

Version 0.2.0

Author Deniz Akdemir [aut, cre]

Maintainer Deniz Akdemir <deniz.akdemir.work@gmail.com>

Description Implements Le Cam deficiency theory for causal inference, as described in Akdemir (2026) <doi:10.5281/zenodo.18367347>. Provides theorem-backed bounds together with computable proxy diagnostics for information loss from confounding, selection bias, and distributional shift. Supports continuous, binary, count, survival, and competing risks outcomes. Key features include propensity-score total-variation deficiency proxies, negative control diagnostics, policy regret bounds, and sensitivity analysis via confounding frontiers.

License MIT + file LICENSE

URL https://github.com/denizakdemir/causaldef

BugReports https://github.com/denizakdemir/causaldef/issues

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 3.6.0)

Imports checkmate, cli, ggplot2, stats, graphics

Suggests survival, MatchIt, tmle, SuperLearner, grf, future.apply, glue, shiny, plumber, jsonlite, cmprsk, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 2

Config/Needs/website pkgdown

VignetteBuilder knitr

Language en-US

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

Date/Publication 2026-03-26 14:13:15 UTC

RemoteUrl https://github.com/denizakdemir/causaldef

RemoteRef HEAD

RemoteSha 9973d26f9475e25906a92ea8409f08667a3543ac

Contents

causaldef-package . . . 3
audit_data . . . 4
causal_spec . . . 6
causal_spec_competing . . . 7
causal_spec_survival . . . 10
confounding_frontier . . . 12
create_plumber_api . . . 14
create_shiny_app_files . . . 15
estimate_deficiency . . . 15
estimate_deficiency_competing . . . 17
estimate_effect . . . 18
frontdoor_effect . . . 19
gene_perturbation . . . 21
hct_outcomes . . . 22
iv_effect . . . 22
nc_diagnostic . . . 24
nsw_benchmark . . . 26
overlap_diagnostic . . . 27
partial_id_set . . . 28
plot.causal_effect . . . 29
plot.confounding_frontier . . . 30
plot.deficiency . . . 30
plot.nc_diagnostic_sensitivity . . . 31
plot.policy_bound . . . 31
plot.transport_deficiency . . . 32
policy_regret_bound . . . 33
policy_regret_bound_vc . . . 35
print.causal_effect . . . 36
print.causal_spec . . . 36
print.causal_spec_competing . . . 37
print.causal_spec_survival . . . 37
print.confounding_frontier . . . 38
print.data_audit_report . . . 38
print.deficiency . . . 39
print.frontdoor_effect . . . 39
print.iv_effect . . . 40
print.nc_diagnostic . . . 40
print.overlap_diagnostic . . . 41

print.partial_id_set . . . 41
print.policy_bound . . . 42
print.transport_deficiency . . . 42
rhc . . . 43
rkhs_rate_bound . . . 43
run_causaldef_api . . . 44
run_causaldef_app . . . 45
sharp_lower_bound . . . 46
summary.iv_effect . . . 46
test_instrument . . . 47
transport_deficiency . . . 48
validate_causal_spec . . . 49
wasserstein_deficiency_gaussian . . . 50

Index **51**

`causaldef-package` *causaldef: Decision-Theoretic Causal Diagnostics via Le Cam Deficiency*

Description

Theory-forward causal diagnostics organized around Le Cam deficiency.

Details

The package centers causal inference around the question: how much information separates the observational study at hand from the interventional experiment we would ideally like to run?

Scientific contract

The package exposes four kinds of quantities:

- **Theorem-backed utilities**: closed-form or theorem-aligned bounds such as `policy_regret_bound()`, `policy_regret_bound_vc()`, `confounding_frontier()`, `sharp_lower_bound()`, and `wasserstein_deficiency_g`
- **Computable deficiency proxies**: `estimate_deficiency()` currently returns a propensity-score total-variation proxy (`metric = "ps_tv"`), not a generic nonparametric estimator of the exact Le Cam deficiency.
- **Sensitivity diagnostics**: functions such as `nc_diagnostic()` combine observable diagnostics with user-supplied sensitivity parameters.
- **Experimental heuristics**: some modules currently expose effect estimates together with heuristic risk scores or proxy summaries. These should be read as diagnostics unless the individual help page states a theorem-backed identification result.

Recommended workflow

1. Specify the causal problem with `causal_spec()` or a survival variant.
2. Compute a pre-specified diagnostic or proxy with `estimate_deficiency()` or a specialized module.
3. Stress-test assumptions with sensitivity tools such as `nc_diagnostic()` or `confounding_frontier()`.
4. Translate the resulting information gap into decision consequences with `policy_regret_bound()`.

Author(s)

Maintainer: Deniz Akdemir <deniz.akdemir.work@gmail.com>

References

Akdemir, D. (2026). Constraints on Causal Inference as Experiment Comparison: A Framework for Identification, Transportability, and Policy Learning. DOI: 10.5281/zenodo.18367347

Le Cam, L., & Yang, G. L. (2000). Asymptotics in Statistics: Some Basic Concepts. Springer.

See Also

Useful links:

- https://github.com/denizakdemir/causaldef
- Report bugs at https://github.com/denizakdemir/causaldef/issues

---

`audit_data` *Audit Data for Causal Validity*

Description

Automatically scans a dataset for causal validity issues using negative control diagnostics (manuscript `thm:nc_bound`). Identifies potential confounders, instruments, and negative control variables.

Usage

```
audit_data(
  data,
  treatment,
  outcome,
  covariates = NULL,
  negative_controls = NULL,
  alpha = 0.05,
  verbose = TRUE
)
```

Arguments

`data`	A data.frame containing the analysis data
`treatment`	Character: name of the treatment variable
`outcome`	Character: name of the outcome variable
`covariates`	Character vector: specific covariates to audit (NULL = all non-treatment/outcome columns)
`negative_controls`	Character vector: known negative control outcomes to validate
`alpha`	Numeric: significance level for tests (default 0.05)
`verbose`	Logical: print progress messages

Details

The auditor tests each variable for:

1. Independence from treatment (correlation test)
2. Correlation with outcome

Based on these tests, variables are classified as:

Potential Instrument Correlates with treatment but NOT outcome

Confounder Correlates with BOTH treatment and outcome

Safe Covariate No significant correlations

If negative_controls are specified, they are validated to ensure they do not show spurious treatment effects (negative control logic).

Value

Object of class "data_audit_report" containing:

- issues: data.frame with columns (variable, issue_type, p_value, recommendation)
- recommendations: character vector of action items
- summary_stats: list with n_vars_audited, n_issues, etc.

References

Akdemir, D. (2026). Constraints on Causal Inference as Experiment Comparison. DOI: 10.5281/zenodo.18367347. See `thm:nc_bound` (Negative Control Sensitivity Bound).

See Also

`nc_diagnostic()`, `causal_spec()`

Examples

```
# Create sample data with known structure
n <- 300
U <- rnorm(n)
W <- U + rnorm(n, sd = 0.5)  # Confounder
Z <- rnorm(n)                 # Potential instrument
A <- rbinom(n, 1, plogis(0.5 * Z + 0.3 * U))
Y <- 2 * A + 1.5 * U + rnorm(n)
df <- data.frame(W = W, Z = Z, A = A, Y = Y)

# Run audit
report <- audit_data(df, treatment = "A", outcome = "Y")
print(report)
```

causal_spec *Create a Causal Problem Specification*

Description

Defines the causal inference problem including treatment, outcome, covariates, and optional diagnostic variables.

Usage

```
causal_spec(
  data,
  treatment,
  outcome,
  covariates = NULL,
  negative_control = NULL,
  instrument = NULL,
  estimand = c("ATE", "ATT", "ATC"),
  outcome_type = c("continuous", "binary", "count"),
  na.action = na.omit
)
```

Arguments

data	A data.frame or data.table containing the analysis data
treatment	Character: name of the treatment variable
outcome	Character: name of the outcome variable
covariates	Character vector: names of adjustment covariates (NULL for none)
negative_control	Character: name of negative control outcome (optional)
instrument	Character: name of instrumental variable (optional)
estimand	Character: target estimand, one of "ATE", "ATT", or "ATC"
outcome_type	Character: type of outcome, one of "continuous", "binary", "count"
na.action	Function: how to handle missing values (default: na.omit)

Value

Object of class "causal_spec" containing:

- data: the analysis dataset after applying na.action to the required variables
- treatment: name of the treatment variable
- outcome: name of the outcome variable
- covariates: adjustment covariates used to define the causal problem
- negative_control: optional negative control outcome name

- `instrument`: optional instrument name
- `estimand`: target causal estimand
- `outcome_type`: declared outcome scale
- `treatment_type`: inferred treatment scale (`"binary"`, `"categorical"`, or `"continuous"`)
- `n`: retained sample size after preprocessing

This object is the package's core problem specification: it records which columns define the observational causal study and is consumed by downstream estimators and diagnostics.

Note: while `causal_spec()` can describe categorical or continuous treatments, the current downstream estimation functions (`estimate_deficiency()`, `estimate_effect()`) require a binary treatment.

See Also

`estimate_deficiency()`, `nc_diagnostic()`, `policy_regret_bound()`

Examples

```
# Create sample data
n <- 200
W <- rnorm(n)
A <- rbinom(n, 1, plogis(0.5 * W))
Y <- 1 + 2 * A + W + rnorm(n)
df <- data.frame(W = W, A = A, Y = Y)

# Create causal specification
spec <- causal_spec(
  data = df,
  treatment = "A",
  outcome = "Y",
  covariates = "W"
)

print(spec)
```

---

causal_spec_competing *Causal Specification for Competing Risks*

Description

Creates a specification object for time-to-event data with competing events. Extends standard survival analysis to handle multiple event types.

Usage

```
causal_spec_competing(
  data,
  treatment,
  time,
  event,
  covariates = NULL,
  event_of_interest = 1,
  horizon = NULL,
  estimand = c("cif", "cshr")
)
```

Arguments

`data`	Data frame containing the variables
`treatment`	Character: name of the treatment variable
`time`	Character: name of the time-to-event variable
`event`	Character: name of the event indicator. Accepts either: • a numeric code with `0` = `censored` and `1,2,...` indicating event types, or • a factor/character (e.g., `"Death"`, `"Relapse"`, `"Censored"`), which will be mapped to integer codes with `"Censored"` (case-insensitive) and `"0"` treated as censored.
`covariates`	Character vector: names of covariate variables
`event_of_interest`	Integer or character: which event type is the primary outcome (default 1). If `event` is factor/character, you may pass the label (e.g., `"Death"`).
`horizon`	Numeric: time horizon for cumulative incidence
`estimand`	Character: "cif" (cumulative incidence) or "cshr" (cause-specific HR)

Details

In competing risks, standard survival methods can be biased because:

1. Censoring the competing event underestimates cumulative incidence
2. Cause-specific hazards don't translate directly to probabilities

The Aalen-Johansen estimator handles this by modeling all transitions:

$$F_k(t) = P(T \leq t, J = k) = \int_0^t S(u-)\lambda_k(u)du$$

Value

Object of class `c("causal_spec_competing", "causal_spec_survival", "causal_spec")` containing:

- `data`: the analysis dataset, with the event column normalized to integer codes (`0` = censored, `1,2,...` = event types)

- `treatment`, `time`, `event`: names of the core analysis variables
- `covariates`: adjustment covariates
- `event_of_interest`: integer code of the primary event type
- `event_types`: observed non-censor event codes
- `n_events`: number of distinct non-censor event types
- `event_map`: optional mapping from original factor/character labels to integer event codes
- `event_levels`: original non-censor event labels when the input event variable was character or factor
- `horizon`: target time horizon for cumulative incidence summaries
- `estimand`: requested competing-risks estimand (`"cif"` or `"cshr"`)
- `treatment_type`: inferred treatment scale
- `n`: retained sample size
- `outcome_type`: fixed label `"competing_risks"`

This object records how the competing-risks problem is encoded and is the input consumed by competing-risks deficiency estimation.

See Also

`estimate_deficiency()`, `causal_spec_survival()`

Examples

```
# Simulate competing risks: death (1) vs transplant (2)
set.seed(42)
n <- 500
W <- rnorm(n)
A <- rbinom(n, 1, plogis(0.3 * W))

# Event times
rate_death <- exp(-0.5 * A + 0.2 * W)
rate_transplant <- exp(0.3 * A - 0.1 * W)

time_death <- rexp(n, rate_death)
time_transplant <- rexp(n, rate_transplant)
time_censor <- runif(n, 0, 3)

observed_time <- pmin(time_death, time_transplant, time_censor)
event <- ifelse(observed_time == time_death, 1,
                ifelse(observed_time == time_transplant, 2, 0))

df <- data.frame(W = W, A = A, time = observed_time, event = event)

spec <- causal_spec_competing(
  df, "A", "time", "event", "W",
  event_of_interest = 1,
  horizon = 2
)
```

```
print(spec)
```

`causal_spec_survival` *Create a Survival Causal Specification*

Description

Defines a causal inference problem with time-to-event outcomes.

Usage

```
causal_spec_survival(
  data,
  treatment,
  time,
  event,
  covariates = NULL,
  competing_event = NULL,
  estimand = c("ATE", "RMST", "HR"),
  horizon = NULL,
  na.action = na.omit
)
```

Arguments

`data`	A data.frame containing survival data
`treatment`	Character: name of the treatment variable
`time`	Character: name of the time-to-event variable
`event`	Character: name of the event indicator (1 = event, 0 = censored). If the column is a factor/character (e.g., `"Death"`, `"Relapse"`, `"Censored"`), it will be coerced to a 0/1 indicator by treating `"Censored"` (case-insensitive) and `"0"` as censored and all other non-missing values as events. For competing risks, prefer `causal_spec_competing()`.
`covariates`	Character vector: names of adjustment covariates
`competing_event`	Character: name of competing event indicator (optional)
`estimand`	Character: target estimand ("ATE", "RMST", "HR")
`horizon`	Numeric: time horizon for RMST (required if estimand = "RMST")
`na.action`	Function: how to handle missing values

Details

For survival outcomes, the deficiency measures how well we can simulate the interventional survival distribution from observational data.

Value

Object of class `c("causal_spec_survival", "causal_spec")` containing:

- `data`: the survival analysis dataset after applying `na.action` and coercing the event indicators when needed
- `treatment`: name of the treatment variable
- `time`: name of the follow-up time variable
- `event`: name of the primary event indicator
- `covariates`: adjustment covariates
- `competing_event`: optional competing-event indicator used by survival workflows that model it separately
- `estimand`: target survival estimand (`"ATE"`, `"RMST"`, or `"HR"`)
- `horizon`: RMST horizon when relevant
- `treatment_type`: inferred treatment scale
- `n`: retained sample size
- `n_events`: number of observed primary events
- `max_time`: maximum observed follow-up time

The returned object defines a time-to-event causal problem for downstream deficiency estimation and effect estimation.

Estimands

ATE Average treatment effect on survival probability at horizon

RMST Restricted mean survival time difference

HR Hazard ratio (log scale)

See Also

`causal_spec()`, `estimate_deficiency()`

Examples

```
# Simulate survival data
n <- 200
W <- rnorm(n)
A <- rbinom(n, 1, plogis(0.5 * W))
time <- rexp(n, rate = exp(-0.5 * A + 0.3 * W))
event <- rbinom(n, 1, 0.8)
df <- data.frame(W = W, A = A, time = time, event = event)

spec <- causal_spec_survival(
  data = df,
  treatment = "A",
  time = "time",
  event = "event",
```

```
  covariates = "W",
  estimand = "RMST",
  horizon = 5
)
```

`confounding_frontier` *Map the Confounding Frontier*

Description

Computes deficiency as a function of confounding strengths (α, γ) for linear Gaussian models.

Usage

```
confounding_frontier(
  spec = NULL,
  alpha_range = c(-2, 2),
  gamma_range = c(-2, 2),
  grid_size = 25,
  model = "gaussian"
)
```

Arguments

`spec`	A causal_spec object (optional, for parameter estimation)
`alpha_range`	Numeric vector c(min, max): range of U->A confounding strength
`gamma_range`	Numeric vector c(min, max): range of U->Y confounding strength
`grid_size`	Integer: resolution per dimension
`model`	Character: model type ("gaussian" for closed-form)

Details

For the linear Gaussian structural causal model:

$$U \sim N(0,1), \quad A = \alpha U + \varepsilon_A, \quad Y = \beta A + \gamma U + \varepsilon_Y$$

The manuscript's Confounding Lower Bound theorem (`thm:confounding_lb`) states the scaling of the causal deficiency:

$$\delta \geq C \frac{|\alpha\gamma|}{\sqrt{(1+\alpha^2/\sigma_A^2)(1+\gamma^2/\sigma_Y^2)}}$$

for some universal constant $C > 0$ (typically after rescaling so $|a| \leq 1$). This function computes a concrete, rigorous lower bound using the underlying Le Cam two-point construction (i.e., $\delta \geq (1/2)\,\mathrm{TV}$ between two observationally indistinguishable parameterizations).

This creates a "confounding frontier" - the boundary where identification transitions from possible ($\delta = 0$) to impossible ($\delta > 0$).

Value

Object of class "confounding_frontier" containing:

- grid: Data frame with columns alpha, gamma, delta
- frontier: Subset of grid where delta ≈ 0
- model: Model type used
- params: Parameters used in computation

Interpretation

α Strength of confounding path U -> A

γ Strength of confounding path U -> Y

$|\alpha\gamma|$ Product determines the confounding bias

When alpha=0 OR gamma=0, the confounder has no effect and $\delta = 0$.

References

Akdemir, D. (2026). Constraints on Causal Inference as Experiment Comparison. DOI: 10.5281/zenodo.18367347. See `thm:confounding_lb` (Confounding Lower Bound).

See Also

`estimate_deficiency()`, `policy_regret_bound()`

Examples

```
# Basic frontier
frontier <- confounding_frontier(
  alpha_range = c(-2, 2),
  gamma_range = c(-2, 2),
  grid_size = 25
)

# With data-based parameter estimation
df <- data.frame(A = rnorm(100), Y = rnorm(100), W = rnorm(100))
spec <- causal_spec(df, "A", "Y", "W")
frontier <- confounding_frontier(spec, grid_size = 50)
```

`create_plumber_api` *Create Plumber API for CausalDef*

Description

Generates a plumber API definition file for deploying causaldef as a REST service. This enables integration with web applications, dashboards, and other programming languages.

Usage

```
create_plumber_api(path = "causaldef_api", port = 8080)
```

Arguments

`path`	Directory to write the API files
`port`	Integer: default port for the API (default 8080)

Details

The API provides the following endpoints:

- `POST /deficiency`: Estimate deficiency from uploaded data
- `POST /policy-bound`: Calculate policy regret bounds
- `POST /frontier`: Generate confounding frontier
- `GET /health`: Health check endpoint
- `GET /methods`: List available methods

Value

Invisibly returns the path to the created files

See Also

`run_causaldef_app()`

Examples

```
## Not run:
# Create API files
create_plumber_api("my_api")

# Run the API
plumber::plumb("my_api/plumber.R")$run(port = 8080)

## End(Not run)
```

create_shiny_app_files *Create Standalone Shiny App Files*

Description

Generates app.R and global.R files for standalone deployment (e.g., shinyapps.io, Shiny Server)

Usage

```
create_shiny_app_files(path = "causaldef_app")
```

Arguments

path	Directory to write the app files

Value

Invisibly returns path, the directory where the generated standalone app files were written. In the current implementation this is the folder containing the generated app.R, which can then be deployed with Shiny hosting tools.

estimate_deficiency *Estimate a Deficiency Proxy (PS-TV)*

Description

Computes a *computable proxy* for the (population) Le Cam deficiency between observational and interventional regimes under various adjustment strategies.

Usage

```
estimate_deficiency(
  spec,
  methods = c("iptw", "aipw"),
  treatment_value = NULL,
  n_boot = 200,
  ci_level = 0.95,
  verbose = TRUE,
  parallel = FALSE
)
```

Arguments

`spec`	A causal_spec object
`methods`	Character vector: adjustment methods to compare • "unadjusted": No adjustment (baseline) • "iptw": Inverse probability weighting • "aipw": Augmented IPW (doubly robust) • "tmle": Targeted maximum likelihood estimation (requires tmle package) • "matching": Propensity score matching (requires MatchIt package) • "grf": Generalized random forests (requires grf package) • "cox_iptw": Cox model with IPTW (survival outcomes, requires survival and a compatible R runtime)
`treatment_value`	Numeric: intervention value for do(A=a)
`n_boot`	Integer: bootstrap replicates for CI (0 = none)
`ci_level`	Numeric: confidence level (default 0.95)
`verbose`	Logical: show progress
`parallel`	Logical: use parallel processing for bootstrap (requires future.apply)

Details

The (population) Le Cam deficiency is defined as:

$$\delta(\mathcal{E}_{obs}, \mathcal{E}_{do}) = \inf_{K} \sup_{\theta} ||KP_{\theta}^{obs} - P_{\theta}^{do}||_{TV}$$

This quantity is not directly estimable nonparametrically from finite samples without additional structure. The current implementation returns a proxy based on the total variation distance between *propensity-score distributions* under:

1. the method-induced reweighting (pseudo-population), and
2. the unweighted target population.

This PS-TV proxy is an overlap/balance diagnostic: values near 0 indicate near-perfect balance under the estimated propensity model, while large values flag positivity/overlap issues.

Importantly, this is a *proxy* and should not be interpreted as the exact Le Cam deficiency unless supplemented by a result linking the proxy to the specific decision class of interest.

Value

Object of class "deficiency" containing:

- estimates: Named vector of deficiency proxy estimates per method
- se: Standard errors (if n_boot > 0)
- ci: Confidence intervals
- method: Methods used
- kernel: Fitted kernels for each method
- metric: Identifier for the proxy metric used

See Also

causal_spec(), nc_diagnostic(), policy_regret_bound()

Examples

```
# Create sample data
n <- 200
W <- rnorm(n)
A <- rbinom(n, 1, plogis(0.5 * W))
Y <- 1 + 2 * A + W + rnorm(n)
df <- data.frame(W = W, A = A, Y = Y)

spec <- causal_spec(df, "A", "Y", "W")
results <- estimate_deficiency(spec, methods = c("unadjusted", "iptw"), n_boot = 50)
print(results)
```

---

estimate_deficiency_competing

Estimate Deficiency for Competing Risks

Description

Computes Le Cam deficiency for competing risks outcomes using cause-specific or subdistribution hazard approaches.

Usage

```
estimate_deficiency_competing(
  spec,
  method = c("cshr", "fg"),
  n_boot = 100,
  ci_level = 0.95
)
```

Arguments

spec	A causal_spec_competing object
method	Character: "cshr" (cause-specific) or "fg" (Fine-Gray subdistribution)
n_boot	Integer: bootstrap replicates
ci_level	Numeric: confidence level

Value

Object of class `c("deficiency_competing", "deficiency")` containing:

- `estimates`: named vector with the competing-risks deficiency proxy estimate
- `se`: bootstrap standard error for that estimate
- `ci`: bootstrap confidence interval matrix
- `method`: the competing-risks estimation strategy used
- `cif`: weighted and unweighted cumulative-incidence calculations by event type
- `spec`: the original `causal_spec_competing` object
- `kernel`: fitted nuisance quantities used by the estimator, including propensity scores and weights

The reported deficiency summarizes how different the weighted and unweighted cumulative-incidence behavior is for the event-of-interest analysis under the selected competing-risks approach.

`estimate_effect` *Estimate Causal Effects from Deficiency Objects*

Description

Uses the fitted causal kernels (from `estimate_deficiency`) to estimate the causal effect (ATE, RMST, etc.).

Usage

```
estimate_effect(object, ...)

## S3 method for class 'deficiency'
estimate_effect(
  object,
  target_method = NULL,
  method = NULL,
  contrast = NULL,
  ...
)
```

Arguments

`object`	A `deficiency` object returned by `estimate_deficiency()`
`...`	Additional arguments passed to specific estimators
`target_method`	Deprecated alias for `method`.
`method`	Character: which adjustment method to use. If `NULL`, defaults to the method with the lowest estimated deficiency.
`contrast`	Vector: c(treated, control) values for contrast. Defaults to c(treatment_value, control_value).

Value

An object of class "causal_effect". Depending on the underlying estimand, the object contains:

- estimate: scalar causal contrast when a point effect is computed
- mu1, mu0: adjusted outcome means for treated and control groups in standard outcome settings
- rmst1, rmst0: restricted mean survival times under the two contrast levels for RMST analyses
- fit: a fitted survival::survfit object when survival curves are returned
- horizon: RMST horizon when relevant
- type: human-readable label for the returned estimand
- method: adjustment method used to construct the effect estimate
- contrast: treated and control values used in the comparison

The object represents the causal effect implied by the fitted kernel stored in the deficiency object for the chosen method.

Examples

```
set.seed(42)
n <- 200
W <- rnorm(n)
A <- rbinom(n, 1, plogis(0.5 * W))
Y <- 1 + 2 * A + W + rnorm(n)
df <- data.frame(W = W, A = A, Y = Y)
spec <- causal_spec(df, "A", "Y", "W")
def <- estimate_deficiency(spec, methods = "iptw")
effect <- estimate_effect(def)
print(effect)
```

---

frontdoor_effect *Front-Door Adjustment Kernel*

Description

Implements the front-door criterion for causal identification when a mediator blocks all paths from treatment to outcome while being unaffected by confounders.

Usage

```
frontdoor_effect(
  spec,
  mediator,
  method = c("plugin", "dr"),
  n_boot = 200,
  ci_level = 0.95
)
```

Arguments

`spec`	A causal_spec object with mediator specified
`mediator`	Character: name of the mediator variable M
`method`	Character: estimation method for the front-door formula • "plugin": Simple plug-in estimator • "dr": Reserved for a future doubly robust front-door estimator. The current release errors if requested.
`n_boot`	Integer: bootstrap replicates for standard errors (default 200)
`ci_level`	Numeric: confidence level (default 0.95)

Details

The front-door criterion (Pearl, 1995) identifies causal effects through a mediator M when:

1. A -> M is unconfounded (M intercepts all directed paths from A to Y)
2. M -> Y has no direct A effect except through M
3. There's no unblocked back-door path from M to Y

The front-door formula is:

$$P(Y|do(a)) = \sum_m P(M = m|A = a) \sum_{a'} P(Y|M = m, A = a')P(A = a')$$

When these assumptions hold, the manuscript's Front-Door Kernel Existence theorem (`thm:frontdoor`) implies $\delta = 0$.

The current implementation estimates the front-door effect, but the reported `deficiency_proxy` is *not* an estimator of the exact Le Cam deficiency. It is a heuristic discrepancy score meant for exploratory diagnostics.

Value

Object of class "frontdoor_effect" containing:

- estimate: The front-door causal effect
- se: Standard error (if bootstrap > 0)
- ci: Confidence interval
- deficiency_proxy: Heuristic proxy based on disagreement between the naive and front-door effect estimates
- deficiency: Backward-compatible alias of `deficiency_proxy`

References

Pearl, J. (1995). Causal diagrams for empirical research. Biometrika. Akdemir, D. (2026). Constraints on Causal Inference as Experiment Comparison. See `thm:frontdoor` (Front-Door Kernel Existence).

See Also

estimate_deficiency(), causal_spec()

Examples

```
# Simulate front-door scenario
n <- 500
U <- rnorm(n)  # Unmeasured confounder
A <- rbinom(n, 1, plogis(0.5 * U))
M <- 0.5 + 1.2 * A + rnorm(n, sd = 0.5)  # Mediator (unconfounded by U)
Y <- 1 + 0.8 * M + 0.5 * U + rnorm(n)    # Outcome
df <- data.frame(A = A, M = M, Y = Y)

spec <- causal_spec(df, "A", "Y", covariates = NULL)

fd_result <- frontdoor_effect(spec, mediator = "M")
print(fd_result)
```

---

gene_perturbation *Gene Expression Perturbation Data*

Description

Simulated high-throughput screening data with potential batch effects. Contains a negative control outcome ("housekeeping_gene").

Usage

```
data(gene_perturbation)
```

Format

A data frame with 500 rows and 6 variables:

sample_id Sample identifier

batch Experimental batch (1-4)

library_size Total read count

knockout_status Intervention (Control vs Knockout)

target_expression Expression level of target gene (outcome)

housekeeping_gene Expression of unrelated housekeeping gene (negative control)

Details

Used to demonstrate negative control diagnostics. The housekeeping gene shares batch effects and library size confounders with the target gene but is not biologically affected by the knockout.

hct_outcomes *Hematopoietic Cell Transplantation Outcomes*

Description

Simulated survival data comparing conditioning intensities in HCT. Illustrates competing risks (Relapse vs Death).

Usage

```
data(hct_outcomes)
```

Format

A data frame with 800 rows and 7 variables:

id Patient ID

age Patient age in years

disease_status Disease stage (Early, Intermediate, Advanced)

kps Karnofsky Performance Score (0-100)

donor_type HLA matching status

conditioning_intensity Treatment group (Myeloablative vs Reduced)

time_to_event Time to first event or censoring

event_status Type of event (Death, Relapse, Censored)

Details

This dataset mimics a retrospective registry study where treatment assignment is confounded by patient health status (e.g., sicker patients get Reduced intensity).

iv_effect *Instrumental Variable Effect Estimation*

Description

Estimates causal effects using instrumental variables (IV) together with first-stage strength diagnostics.

Usage

```
iv_effect(
  spec,
  instrument = NULL,
  method = c("2sls", "wald", "liml"),
  n_boot = 200,
  ci_level = 0.95,
  weak_iv_threshold = 10
)
```

Arguments

`spec`	A causal_spec object with instrument specified
`instrument`	Character: name of the instrumental variable (or use spec$instrument)
`method`	Character: IV estimation method • "2sls": Two-stage least squares • "wald": Wald estimator (ratio of reduced forms) • "liml": Reserved for a future limited-information maximum likelihood implementation. The current release errors if requested.
`n_boot`	Integer: bootstrap replicates for CI
`ci_level`	Numeric: confidence level
`weak_iv_threshold`	Numeric: F-statistic threshold for weak IV (default 10)

Details

Instrumental variables identify the Local Average Treatment Effect (LATE) for compliers when:

1. Relevance: Z affects A (testable via F-stat)
2. Exclusion: Z affects Y only through A (untestable)
3. Independence: Z is independent of unmeasured confounders

Weak instruments (small first-stage F-statistics) lead to unstable IV inference. The current implementation reports a `deficiency_proxy` derived from instrument strength; it should be interpreted as a screening diagnostic rather than an exact Le Cam deficiency.

Value

Object of class "iv_effect" containing:

- estimate: The IV causal effect (LATE)
- se: Standard error
- ci: Confidence interval
- f_stat: First-stage F-statistic
- weak_iv: Logical, whether instrument is weak
- deficiency_proxy: Heuristic proxy derived from first-stage strength
- deficiency: Backward-compatible alias of `deficiency_proxy`

References

Angrist, J. D., Imbens, G. W., & Rubin, D. B. (1996). Identification of causal effects using instrumental variables. JASA.

See Also

causal_spec(), estimate_deficiency()

Examples

```
# Simulate IV setting
set.seed(42)
n <- 1000
U <- rnorm(n)  # Unmeasured confounder
Z <- rbinom(n, 1, 0.5)  # Instrument (randomized encouragement)
A <- 0.3 + 0.4 * Z + 0.3 * U + rnorm(n, sd = 0.3)  # Treatment (continuous)
A <- as.numeric(A > 0.5)  # Dichotomize
Y <- 1 + 2 * A + 0.8 * U + rnorm(n)  # Outcome

df <- data.frame(Z = Z, A = A, Y = Y)
spec <- causal_spec(df, "A", "Y", instrument = "Z")


iv_result <- iv_effect(spec)
print(iv_result)
```

---

nc_diagnostic *Negative Control Diagnostic*

Description

Screens for residual association with a negative control outcome and computes the corresponding κ-based sensitivity bound.

Usage

```
nc_diagnostic(
  spec,
  method = "iptw",
  kappa = NULL,
  kappa_range = NULL,
  alpha = 0.05,
  n_boot = 200
)
```

Arguments

spec	A causal_spec with negative_control specified
method	Character: adjustment method to test
kappa	Numeric: alignment constant in the theoretical bound. If NULL, defaults to 1 (conservative).
kappa_range	Numeric vector: range of kappa values for sensitivity analysis. If provided, returns bounds for each kappa and adds class "nc_diagnostic_sensitivity".
alpha	Numeric: significance level for the screening test
n_boot	Integer: number of permutation draws used for the screening test

Details

The diagnostic uses the bound: $\delta(\hat{K}) \leq \kappa \cdot \delta_{NC}(\hat{K})$, where δ_{NC} is the negative control deficiency.

A negative control outcome Y' is a variable that:

1. Shares confounders with Y (affected by U)
2. Is NOT affected by treatment A

If adjustment correctly removes confounding, the residual association between A and Y' should be zero. Non-zero association indicates confounding remains.

The current implementation separates two tasks:

1. a **screening test** based on a weighted correlation statistic and a permutation null distribution, and
2. a **sensitivity bound** of the form $\delta \leq \kappa\delta_{NC}$ based on the observed residual-association proxy delta_nc.

This keeps the observable screening step distinct from the theorem's user-supplied alignment parameter kappa.

When kappa is unknown, use kappa_range to perform sensitivity analysis across plausible values. The resulting plot shows how the deficiency bound varies with the sensitivity parameter.

Value

Object of class "nc_diagnostic" containing:

- delta_nc: Observable residual-association proxy based on a weighted treatment/negative-control association
- delta_bound: Upper bound on true deficiency (kappa * delta_nc)
- falsified: Logical indicating if assumptions are falsified
- p_value: Permutation p-value for the observed residual association
- kappa: Alignment constant used
- screening: List with the observed association statistic, permutation p-value, and null standard deviation
- sensitivity: (if kappa_range provided) data.frame with bounds for each kappa

References

Akdemir, D. (2026). Constraints on Causal Inference as Experiment Comparison. DOI: 10.5281/zenodo.18367347. See thm:nc_bound (Negative Control Sensitivity Bound).

Lipsitch, M., Tchetgen, E., & Cohen, T. (2010). Negative controls: A tool for detecting confounding and bias. Epidemiology, 21(3), 383-388.

See Also

causal_spec(), estimate_deficiency()

Examples

```
# Create data with negative control
n <- 200
U <- rnorm(n)
W <- U + rnorm(n, sd = 0.5)
A <- rbinom(n, 1, plogis(0.5 * W))  # Binary treatment
Y <- 1 + 2 * A + U + rnorm(n)
Y_nc <- U + rnorm(n)  # Shares U but no effect from A
df <- data.frame(W = W, A = A, Y = Y, Y_nc = Y_nc)

spec <- causal_spec(df, "A", "Y", "W", negative_control = "Y_nc")

# Standard diagnostic
nc_result <- nc_diagnostic(spec, method = "iptw", n_boot = 50)
print(nc_result)

# Sensitivity analysis
nc_sens <- nc_diagnostic(spec, method = "iptw",
                         kappa_range = seq(0.5, 2.0, by = 0.25),
                         n_boot = 50)
plot(nc_sens)
```

---

nsw_benchmark *Lalonde National Supported Work (NSW) Benchmark*

Description

The classical causal inference benchmark dataset constructed by Dehejia and Wahba (1999) from the original Lalonde (1986) study. This version combines the experimental NSW groups with the observational CPS and PSID comparison groups, enabling precise calculation of the "deficiency" between observational and experimental inference.

Usage

```
data(nsw_benchmark)
```

Format

A data frame with 2915 rows and 11 variables:

treat Treatment indicator (1 = Job Training, 0 = Control)

age Age in years

education Years of schooling

black Indicator for Black race

hispanic Indicator for Hispanic race

married Indicator for marital status

nodegree Indicator for no high school degree

re74 Real earnings in 1974 (pre-treatment)

re75 Real earnings in 1975 (pre-treatment)

re78 Real earnings in 1978 (outcome)

sample_id Source of the observation: "nsw_treated", "nsw_control", "cps_control", or "psid_control".

Details

This dataset allows you to verify causal methods by using the experimental subset ("nsw_treated" vs "nsw_control") as the ground truth, and comparing the results obtained by adjusting the observational subsets ("nsw_treated" vs "cps_control").

References

LaLonde, R. J. (1986). Evaluating the econometric evaluations of training programs with experimental data. The American Economic Review, 604-620.

Dehejia, R. H., & Wahba, S. (1999). Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs. Journal of the American statistical Association, 94(448), 1053-1062.

`overlap_diagnostic` *Overlap Diagnostic*

Description

Summarizes overlap/positivity issues via propensity score distribution and effective sample size under IPTW weights.

Usage

```
overlap_diagnostic(spec, trim = 0.01)
```

Arguments

spec	A causal_spec.
trim	Trimming threshold in (0, 0.5). Values outside $[trim, 1 - trim]$ are flagged as extreme.

Value

An object of class "overlap_diagnostic" containing:

- trim: trimming threshold used to classify extreme propensity scores
- n: sample size
- extreme_n: number of observations with propensity scores outside the acceptable overlap region
- kept_n: number of observations remaining after trimming
- ps_summary: selected propensity-score quantiles
- ess_iptw: effective sample size under IPTW weights

These quantities summarize how much practical overlap is available for weighted causal estimation.

---

partial_id_set *Partial Identification Set from a Delta Radius (Result 8)*

Description

Converts a deficiency radius δ into a conservative identified interval for bounded functionals under TV.

Usage

```
partial_id_set(
  estimate,
  delta,
  estimand = c("ate", "mean"),
  outcome_range = c(0, 1)
)
```

Arguments

estimate	Point estimate of the functional (numeric).
delta	Deficiency radius $\delta \in [0, 1]$.
estimand	One of "mean" or "ate".
outcome_range	Numeric vector c(min, max) bounding the outcome.

Details

For a mean functional with $Y \in [y_{min}, y_{max}]$, TV control yields $|E_Q[Y] - E_{Q_0}[Y]| \leq 2\delta(y_{max} - y_{min})$. For an ATE (difference of two means), the half-width doubles again.

Value

An object of class `"partial_id_set"` containing:

- `estimand`: whether the interval applies to a bounded mean or an ATE
- `estimate`: the supplied point estimate
- `delta`: the deficiency radius used to widen the estimate
- `outcome_range`: assumed outcome bounds
- `half_width`: worst-case expansion implied by `delta`
- `lower`, `upper`: conservative identification limits

The interval describes all values consistent with the supplied estimate and the chosen total-variation deficiency radius.

`plot.causal_effect` *Plot Causal Effect*

Description

Plot Causal Effect

Usage

```
## S3 method for class 'causal_effect'
plot(x, ...)
```

Arguments

`x`	A causal_effect object
`...`	Additional arguments passed to plot functions

Value

If `x` contains a fitted `survival::survfit` object, the function is called for its side effect of drawing base-R survival curves and returns `NULL` invisibly. Otherwise it returns a `ggplot2` object showing the point estimate for `x$estimate` and, when available, its confidence interval.

plot.confounding_frontier *Plot Confounding Frontier*

Description

Plot Confounding Frontier

Usage

```
## S3 method for class 'confounding_frontier'
plot(x, type = c("heatmap", "contour"), threshold = c(0.05, 0.1, 0.2), ...)
```

Arguments

x	A confounding_frontier object
type	Character: plot type ("heatmap", "contour")
threshold	Numeric vector: threshold values for contour lines
...	Additional arguments

Value

A ggplot2 object visualizing the confounding sensitivity surface over the (alpha, gamma) grid. type = "heatmap" returns a tile plot of x$grid$delta with optional contour lines and benchmark points, while type = "contour" returns filled contours of the same surface.

plot.deficiency *Plot Deficiency Estimates*

Description

Plot Deficiency Estimates

Usage

```
## S3 method for class 'deficiency'
plot(x, type = c("bar", "forest"), ...)
```

Arguments

x	A deficiency object
type	Character: plot type ("bar", "forest")
...	Additional arguments passed to ggplot2 functions

Value

A ggplot2 object. With type = "bar", the plot shows the method-specific deficiency estimates as bars with optional confidence intervals and interpretation thresholds. With type = "forest", the plot shows the same estimates horizontally with optional confidence intervals.

---

plot.nc_diagnostic_sensitivity *Plot Negative Control Sensitivity Analysis*

Description

Visualizes how the deficiency bound varies across different values of the sensitivity parameter kappa. This implements the sensitivity analysis approach from the Negative Control Sensitivity Bound in the manuscript (thm:nc_bound).

Usage

```
## S3 method for class 'nc_diagnostic_sensitivity'
plot(x, threshold = 0.05, ...)
```

Arguments

x	An nc_diagnostic_sensitivity object (from nc_diagnostic with kappa_range)
threshold	Numeric: threshold line to add (default 0.05)
...	Additional arguments passed to plot functions

Value

If ggplot2 is available, a ggplot2 object showing how delta_bound varies across kappa values, with an optional threshold line. Otherwise the function is called for its side effect of producing a base-R plot and returns NULL invisibly.

---

plot.policy_bound *Plot Policy Regret Bound*

Description

Plot Policy Regret Bound

Usage

```
## S3 method for class 'policy_bound'
plot(x, type = c("decomposition", "safety_curve"), ...)
```

Arguments

x	A policy_bound object
type	Character: plot type
...	Additional arguments

Value

A ggplot2 object. With type = "decomposition", the plot decomposes the policy-regret bound into observed regret, transfer penalty, and minimax floor components. With type = "safety_curve", the plot shows how the transfer penalty and minimax floor vary as a function of deficiency, with fitted-method points overlaid when available.

---

plot.transport_deficiency

Plot method for transport_deficiency

Description

Plot method for transport_deficiency

Usage

```
## S3 method for class 'transport_deficiency'
plot(x, type = c("shift", "weights"), ...)
```

Arguments

x	A transport_deficiency object
type	Character: type of plot ("shift" or "weights")
...	Additional arguments passed to plot

Value

A ggplot2 object. With type = "shift", the returned plot is a horizontal bar chart of x$covariate_shift, where bar height is the variable-specific shift metric and fill encodes the severity category. With type = "weights", the returned plot is a histogram of x$weights with a reference line at 1 and a subtitle summarizing the effective sample size after weighting.

`policy_regret_bound` *Compute Policy Regret Bounds*

Description

Given deficiency δ, computes worst-case bounds on policy regret. The bound states: $Regret_{do} \leq Regret_{obs} + M \cdot \delta$

Usage

```
policy_regret_bound(
  deficiency,
  utility_range = c(0, 1),
  obs_regret = NULL,
  policy_class = NULL,
  delta_mode = c("point", "upper"),
  method = NULL
)
```

Arguments

`deficiency`	A deficiency object or numeric δ value (between 0 and 1)
`utility_range`	Numeric vector c(min, max) of utility bounds
`obs_regret`	Numeric: observed regret from policy on observational data (optional)
`policy_class`	Character: description of policy class (for reporting)
`delta_mode`	Character: how to extract δ from a deficiency object. • "point": use the minimum point estimate (default) • "upper": use the minimum upper CI bound (more conservative; requires bootstrap)
`method`	Character: when `deficiency` is a `deficiency` object, which fitted method to use when translating the estimate into a regret bound. Strongly recommended whenever multiple methods were compared.

Details

For any policy π learned from observational data, the regret bound is:

$$Regret_{do}(\pi) \leq Regret_{obs}(\pi) + M \cdot \delta$$

The "transfer penalty" $M\delta$ quantifies worst-case regret inflation from the information gap between observational and interventional experiments.

When a multi-method `deficiency` object is supplied, the safest workflow is to pre-specify `method` before calling `policy_regret_bound()`. If `method` is omitted, the current implementation falls back to the smallest available estimate (or upper confidence limit), which is convenient for exploration but optimistic after post-selection.

Separately, a minimax lower bound ("minimax floor") of $(M/2)\delta$ holds for bounded utilities (see manuscript theorem `thm:safety_floor`).

Value

Object of class "policy_bound" containing:

- regret_bound: Upper bound on interventional regret (if obs_regret provided)
- safety_floor: Backward-compatible alias for transfer_penalty
- transfer_penalty: Additive regret penalty = M*δ
- minimax_floor: Minimax lower bound = (M/2)*δ
- delta: Deficiency value used
- delta_selection: How the δ value was selected
- utility_range: Range of utility function
- M: Utility range (max - min)

Implications for Safe AI

If $\delta > 0$ (unobserved confounding exists), no algorithm can guarantee zero regret. The deficiency δ is the "price of safety" for deploying policies without randomized experimentation.

References

Akdemir, D. (2026). Constraints on Causal Inference as Experiment Comparison. DOI: 10.5281/zenodo.18367347. See `thm:policy_regret` (Policy Regret Transfer) and `thm:safety_floor` (Minimax Safety Floor).

See Also

`estimate_deficiency()`, `confounding_frontier()`

Examples

```
# From a deficiency estimate
# From a deficiency estimate
df <- data.frame(W=rnorm(100), A=rbinom(100,1,0.5), Y=rnorm(100))
spec <- causal_spec(df, "A", "Y", "W")
def <- estimate_deficiency(spec, methods = "iptw", n_boot = 0)
bound <- policy_regret_bound(def, utility_range = c(0, 1))

# From a numeric value
bound <- policy_regret_bound(
  deficiency = 0.1,
  utility_range = c(0, 100),
  obs_regret = 5
)
print(bound)
```

policy_regret_bound_vc *Policy Regret Bound with VC Term (Result 3)*

Description

Extends `policy_regret_bound()` with the explicit VC-complexity penalty from Result 3: $CM\sqrt{(VC(\Pi)\log n + \log(1/\xi)}$

Usage

```
policy_regret_bound_vc(
  deficiency,
  vc_dim,
  n,
  xi = 0.05,
  utility_range = c(0, 1),
  obs_regret = NULL,
  delta_mode = c("point", "upper"),
  C = 2
)
```

Arguments

`deficiency`	A `deficiency` object or numeric δ.
`vc_dim`	VC dimension of the policy class.
`n`	Sample size used to learn the policy.
`xi`	Failure probability (ξ).
`utility_range`	Numeric vector c(min, max) of utility bounds.
`obs_regret`	Optional observed regret under the observational regime.
`delta_mode`	Passed through to `policy_regret_bound()` when `deficiency` is an object.
`C`	Universal constant in the bound (default 2).

Value

An object of class `policy_bound` with additional field `complexity_penalty`.

print.causal_effect *Print method for causal_effect*

Description

Print method for causal_effect

Usage

```
## S3 method for class 'causal_effect'
print(x, ...)
```

Arguments

x	A causal_effect object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the estimation method, estimand type, treatment contrast, scalar effect estimate when available, and the horizon for RMST analyses.

print.causal_spec *Print method for causal_spec*

Description

Print method for causal_spec

Usage

```
## S3 method for class 'causal_spec'
print(x, ...)
```

Arguments

x	A causal_spec object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the treatment and outcome variables with their inferred types, the adjustment covariates, sample size, estimand, and any registered negative control or instrument.

print.causal_spec_competing

Print method for causal_spec_competing

Description

Print method for causal_spec_competing

Usage

```
## S3 method for class 'causal_spec_competing'
print(x, ...)
```

Arguments

x	A causal_spec_competing object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed output summarizes the competing-risks specification: treatment, time and event variables, event-of-interest coding, time horizon, estimand, optional event-label mapping, covariates, sample size, and the observed event-count breakdown.

print.causal_spec_survival

Print method for causal_spec_survival

Description

Print method for causal_spec_survival

Usage

```
## S3 method for class 'causal_spec_survival'
print(x, ...)
```

Arguments

x	A causal_spec_survival object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the treatment, follow-up time, event indicator, number of events, covariates, sample size, estimand, and any specified horizon or competing event.

`print.confounding_frontier` *Print method for confounding_frontier*

Description

Print method for confounding_frontier

Usage

```
## S3 method for class 'confounding_frontier'
print(x, ...)
```

Arguments

`x`	A confounding_frontier object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`, unchanged. The printed summary reports the sensitivity-grid dimensions, alpha and gamma ranges, model label, summary statistics of the deficiency surface, and the proportion of the grid with near-zero deficiency.

`print.data_audit_report` *Print method for data_audit_report*

Description

Print method for data_audit_report

Usage

```
## S3 method for class 'data_audit_report'
print(x, ...)
```

Arguments

`x`	A data_audit_report object
`...`	Additional arguments (unused)

Value

Invisibly returns `x`, unchanged. The printed report summarizes the audited treatment/outcome pair, number of variables reviewed, number of flagged issues, a table of non-safe findings, and the recommended next actions.

print.deficiency *Print method for deficiency*

Description

Print method for deficiency

Usage

```
## S3 method for class 'deficiency'
print(x, ...)
```

Arguments

x	A deficiency object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary shows the method-specific deficiency estimates, optional standard errors and confidence intervals, qualitative traffic-light labels, any PS-TV proxy note, and the method with the smallest reported deficiency.

print.frontdoor_effect *Print method for frontdoor_effect*

Description

Print method for frontdoor_effect

Usage

```
## S3 method for class 'frontdoor_effect'
print(x, ...)
```

Arguments

x	A frontdoor_effect object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed output reports the mediator, estimation method, frontdoor effect estimate, optional bootstrap standard error and confidence interval, and the heuristic discrepancy proxy stored in x$deficiency_proxy.

print.iv_effect *Print method for iv_effect*

Description

Print method for iv_effect

Usage

```
## S3 method for class 'iv_effect'
print(x, ...)
```

Arguments

x	An iv_effect object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed output reports the instrument and estimation method, first-stage F-statistic, weak-IV flag, IV effect estimate, optional bootstrap uncertainty, and the heuristic strength proxy stored in x$deficiency_proxy.

print.nc_diagnostic *Print method for nc_diagnostic*

Description

Print method for nc_diagnostic

Usage

```
## S3 method for class 'nc_diagnostic'
print(x, ...)
```

Arguments

x	An nc_diagnostic object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the screening statistic, observed negative-control proxy delta_nc, the translated deficiency bound delta_bound, the chosen kappa, the permutation p-value, and whether the negative-control screen was rejected.

print.overlap_diagnostic *Print method for overlap_diagnostic*

Description

Print method for overlap_diagnostic

Usage

```
## S3 method for class 'overlap_diagnostic'
print(x, ...)
```

Arguments

x	An overlap_diagnostic object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the sample size, trimming threshold, number of extreme propensity scores, number retained after trimming, IPTW effective sample size, and propensity score quantiles.

print.partial_id_set *Print method for partial_id_set*

Description

Print method for partial_id_set

Usage

```
## S3 method for class 'partial_id_set'
print(x, ...)
```

Arguments

x	A partial_id_set object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the target estimand, point estimate, chosen deficiency radius, resulting half-width, and the conservative identification interval implied by that radius.

print.policy_bound *Print method for policy_bound*

Description

Print method for policy_bound

Usage

```
## S3 method for class 'policy_bound'
print(x, ...)
```

Arguments

x	A policy_bound object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the deficiency value used in the regret bound, how that value was selected, the utility range, transfer-penalty and minimax components, any observed regret supplied by the user, and the resulting interpretation on the utility scale.

print.transport_deficiency *Print method for transport_deficiency*

Description

Print method for transport_deficiency

Usage

```
## S3 method for class 'transport_deficiency'
print(x, ...)
```

Arguments

x	A transport_deficiency object
...	Additional arguments (unused)

Value

Invisibly returns x, unchanged. The printed summary reports the transport-risk proxy, optional bootstrap uncertainty, effective sample size under the transport weights, the per-variable covariate-shift table, and the source versus transported ATE estimates.

rhc *Right Heart Catheterization (RHC) Dataset*

Description

Data from the SUPPORT study (Connors et al., 1996) examining the effectiveness of Right Heart Catheterization (RHC) in the management of critically ill patients. This dataset is a classic example of high-dimensional confounding in medical observational studies.

Usage

```
data(rhc)
```

Format

A data frame with 5735 rows and 63 variables.

Details

The original study found that RHC was associated with higher mortality, contradicting potential benefits. Careful adjustment for the rich set of covariates (indicating sickness severity) is required. This dataset serves as a testbed for sensitivity analysis and policy bounds.

Key variables include `swang1` (treatment), `dth30` (30-day mortality), `t3d30` (survival time up to 30 days), and `aps1` (APACHE III score).

References

Connors, A. F., Speroff, T., Dawson, N. V., Thomas, C., Harrell, F. E., Wagner, D., ... & Goldman, L. (1996). The effectiveness of right heart catheterization in the initial care of critically ill patients. JAMA, 276(11), 889-897.

rkhs_rate_bound *RKHS Rate Bound (Result 1)*

Description

Computes the explicit rate bound stated in Result 1 (finite-sample RKHS rate).

Usage

```
rkhs_rate_bound(n, beta, d_w, eta, xi = 0.05)
```

Arguments

n	Sample size.
beta	Smoothness exponent (β in a β-H\"older class).
d_w	Covariate dimension (d_W).
eta	Positivity constant ($\eta > 0$) such that $P(A = a \mid W = w) \geq \eta$.
xi	Failure probability ($\xi \in (0, 1)$).

Value

Numeric rate bound (up to universal constants).

---

run_causaldef_api *Run CausalDef API Server*

Description

Convenience function to start the plumber API server

Usage

```
run_causaldef_api(port = 8080, swagger = TRUE, path = NULL)
```

Arguments

port	Integer: port number (default 8080)
swagger	Logical: enable Swagger UI (default TRUE)
path	Character: path to API files (will create if missing)

Value

This function is called primarily for its side effect of starting a blocking Plumber API server on `port`. No stable structured return value is guaranteed by `causaldef`; when control returns to R, the function returns whatever value is produced by `plumber::plumb(api_file)$run(...)`.

`run_causaldef_app` *Launch CausalDef Shiny Dashboard*

Description

Launches an interactive Shiny application for deficiency analysis. Provides a point-and-click interface for the full causaldef workflow.

Usage

```
run_causaldef_app(data = NULL, port = 3838, launch.browser = TRUE)
```

Arguments

`data`	Optional data frame to preload
`port`	Integer: port for the Shiny server (default 3838)
`launch.browser`	Logical: open in browser? (default TRUE)

Details

The dashboard provides:

- Data upload and variable selection
- Method comparison (unadjusted, IPTW, AIPW, etc.)
- Interactive visualization of deficiency estimates
- Confounding frontier explorer
- Policy regret calculator
- Exportable reports

Value

Invisibly returns the Shiny app object

Examples

```
## Not run:
# Launch with example data
df <- data.frame(
  W = rnorm(200),
  A = rbinom(200, 1, 0.5),
  Y = rnorm(200)
)
run_causaldef_app(data = df)

## End(Not run)
```

sharp_lower_bound *Sharp Two-Point Bounds (Result 4)*

Description

Computes matching lower/upper bounds (sharpness) in the linear Gaussian model, for either TV-based (two-point construction underlying `thm:confounding_lb`) or Wasserstein-1-based (Result 6 two-point construction) deficiencies.

Usage

```
sharp_lower_bound(
  alpha,
  gamma,
  sigma_A = 1,
  sigma_Y = 1,
  a = 1,
  metric = c("tv", "wasserstein")
)
```

Arguments

`alpha`	Confounding strength U -> A.
`gamma`	Confounding strength U -> Y.
`sigma_A`	Treatment noise standard deviation.
`sigma_Y`	Outcome noise standard deviation (TV case).
`a`	Intervention level.
`metric`	One of `"tv"` or `"wasserstein"`.

Value

List with fields: `lower`, `upper`, `ratio`, and `metric`.

summary.iv_effect *Summary method for iv_effect*

Description

Summary method for iv_effect

Usage

```
## S3 method for class 'iv_effect'
summary(object, ...)
```

Arguments

object	An iv_effect object
...	Additional arguments (unused)

Value

Invisibly returns object, unchanged. The printed summary includes the instrument, estimation method, sample size, the coefficient table from the first-stage regression stored in object$first_stage, the IV effect estimate with optional bootstrap uncertainty, and the weak-instrument diagnostics.

---

test_instrument *Test Instrument Validity*

Description

Performs diagnostic tests for instrumental variable assumptions

Usage

```
test_instrument(spec, instrument = NULL, overid_test = TRUE)
```

Arguments

spec	A causal_spec with instrument
instrument	Instrument name
overid_test	Logical: perform overidentification test if multiple instruments

Value

A named list of IV screening diagnostics with components such as:

- relevance: first-stage F-statistic, its p-value, and a logical pass/fail flag relative to the weak-IV threshold of 10
- balance: when covariates are present, per-covariate balance p-values and an overall pass/fail flag for instrument-covariate balance
- reduced_form: coefficient, standard error, p-value, and significance flag for the reduced-form association of the instrument with the outcome

These outputs are diagnostics for testable parts of the IV design. They do not by themselves prove the exclusion restriction or full instrument validity.

`transport_deficiency` *Transport Diagnostic Between Source and Target Populations*

Description

Computes a transport diagnostic for extrapolating causal effects from a source population to a target population.

Usage

```
transport_deficiency(
  source_spec,
  target_data,
  transport_vars = NULL,
  method = c("iptw", "calibration", "sace"),
  n_boot = 100,
  ci_level = 0.95
)
```

Arguments

`source_spec`	A causal_spec object for the source (training) population
`target_data`	Data frame: the target population (may have fewer variables)
`transport_vars`	Character vector: variables that may shift between populations
`method`	Character: method for transport estimation • "iptw": Inverse probability of trial/source weighting • "calibration": Calibration weighting • "sace": Sample average causal effect
`n_boot`	Integer: bootstrap replicates (default 100)
`ci_level`	Numeric: confidence level (default 0.95)

Details

A transport problem can be framed through deficiency, but the current release does *not* estimate an exact transport deficiency. Instead, it returns a heuristic transport-risk proxy that combines observable covariate shift, transport-weight instability, and source/target sample-size imbalance.

Key assumptions for valid transport:

1. Treatment effects are identifiable in source
2. Effect homogeneity or sufficient transport variables
3. Positivity: overlap between source and target covariate distributions

Value

Object of class "transport_deficiency" containing:

- delta_transport_proxy: Heuristic transport-risk proxy combining covariate shift, weight instability, and sample-size imbalance
- delta_transport: Backward-compatible alias of `delta_transport_proxy`
- covariate_shift: Per-variable shift diagnostics
- weights: Transport weights for source observations
- ate_source: ATE estimate in source population
- ate_target: Transported ATE estimate for target

See Also

`estimate_deficiency()`, `policy_regret_bound()`

Examples

```
# Source population (RCT)
set.seed(42)
n_source <- 500
age_s <- rnorm(n_source, 50, 10)
A_s <- rbinom(n_source, 1, 0.5)  # Randomized
Y_s <- 10 + 2 * A_s - 0.1 * age_s + rnorm(n_source)
source_df <- data.frame(age = age_s, A = A_s, Y = Y_s, S = 1)

# Target population (different age distribution)
n_target <- 300
age_t <- rnorm(n_target, 65, 8)  # Older population
target_df <- data.frame(age = age_t, S = 0)

source_spec <- causal_spec(source_df, "A", "Y", "age")

transport <- transport_deficiency(
  source_spec,
  target_data = target_df,
  transport_vars = "age"
)
print(transport)
```

---

validate_causal_spec *Validate Causal Specification*

Description

Validates a causal_spec object

Usage

```
validate_causal_spec(x)
```

Arguments

x	A causal_spec object

Value

The validated object (invisibly)

---

wasserstein_deficiency_gaussian *Wasserstein Deficiency (Linear Gaussian) (Result 6)*

Description

Closed-form Wasserstein-1 deficiency for the two-point construction in Result 6: $\delta_{W_1}(a) = |a\alpha\gamma|/(\alpha^2 + \sigma_A^2)$.

Usage

```
wasserstein_deficiency_gaussian(alpha, gamma, sigma_A = 1, a = 1)
```

Arguments

alpha	Confounding strength U -> A.
gamma	Confounding strength U -> Y.
sigma_A	Treatment noise standard deviation.
a	Intervention level $do(A = a)$.

Value

Numeric deficiency value.

Index

∗ **datasets**
gene_perturbation, 21
hct_outcomes, 22
nsw_benchmark, 26
rhc, 43

audit_data, 4

causal_spec, 6
causal_spec(), *3*, *5*, *11*, *17*, *21*, *24*, *26*
causal_spec_competing, 7
causal_spec_competing(), *10*
causal_spec_survival, 10
causal_spec_survival(), *9*
causaldef *(*causaldef-package*)*, 3
causaldef-package, 3
confounding_frontier, 12
confounding_frontier(), *3*, *34*
create_plumber_api, 14
create_shiny_app_files, 15

estimate_deficiency, 15
estimate_deficiency(), *3*, *7*, *9*, *11*, *13*, *21*, *24*, *26*, *34*, *49*
estimate_deficiency_competing, 17
estimate_effect, 18

frontdoor_effect, 19

gene_perturbation, 21

hct_outcomes, 22

iv_effect, 22

nc_diagnostic, 24
nc_diagnostic(), *3*, *5*, *7*, *17*
nsw_benchmark, 26

overlap_diagnostic, 27

partial_id_set, 28

plot.causal_effect, 29
plot.confounding_frontier, 30
plot.deficiency, 30
plot.nc_diagnostic_sensitivity, 31
plot.policy_bound, 31
plot.transport_deficiency, 32
policy_regret_bound, 33
policy_regret_bound(), *3*, *7*, *13*, *17*, *49*
policy_regret_bound_vc, 35
policy_regret_bound_vc(), *3*
print.causal_effect, 36
print.causal_spec, 36
print.causal_spec_competing, 37
print.causal_spec_survival, 37
print.confounding_frontier, 38
print.data_audit_report, 38
print.deficiency, 39
print.frontdoor_effect, 39
print.iv_effect, 40
print.nc_diagnostic, 40
print.overlap_diagnostic, 41
print.partial_id_set, 41
print.policy_bound, 42
print.transport_deficiency, 42

rhc, 43
rkhs_rate_bound, 43
run_causaldef_api, 44
run_causaldef_app, 45
run_causaldef_app(), *14*

sharp_lower_bound, 46
sharp_lower_bound(), *3*
summary.iv_effect, 46

test_instrument, 47
transport_deficiency, 48

validate_causal_spec, 49

wasserstein_deficiency_gaussian, 50

wasserstein_deficiency_gaussian(), *3*