Package 'causalsim'

August 30, 2026

Title Simulation-Ready Causal Data Generating Processes

Version 0.1.0

Description Construct, simulate, and evaluate causal data generating processes (DGPs) with known ground truth. Designed for benchmarking causal estimators, studying confounding and treatment-effect heterogeneity, and building reproducible teaching examples. Covariate roles (confounder, effect modifier, noise) and heterogeneous treatment effects are first-class concepts in the API, and estimator performance is summarised with bias, root mean squared error, confidence-interval coverage, and power.

License MIT + file LICENSE

URL https://chaycereed.github.io/causalsim/, https://github.com/chaycereed/causalsim

BugReports https://github.com/chaycereed/causalsim/issues

Encoding UTF-8

Depends R (>= 4.0.0)

RoxygenNote 7.3.3

Suggests knitr, pkgdown, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

NeedsCompilation no

Author Chayce Reed [aut, cre]

Maintainer Chayce Reed <Chayce.Reed.HSE@dartmouth.edu>

Repository CRAN

Date/Publication 2026-08-30 10:00:15 UTC

Contents

`causalsim` *Simulate a causal dataset with known ground truth*

Description

A convenience wrapper around `causalsim_dgp()` and `causalsim_draw()`. Defines a causal data generating process and returns one simulated dataset in a single call.

Usage

```
causalsim(
  n,
  effect = 1,
  propensity = "moderate",
  baseline = "moderate",
  sigma = 1,
  covariates = list(),
  n_confounders = 0L,
  n_effect_modifiers = 0L,
  n_noise = 0L,
  mc_draws = 10000L,
  seed = NULL
)
```

Arguments

`n`	Positive integer. Sample size for each simulated dataset.
`effect`	Numeric scalar or function. A scalar specifies a constant (homogeneous) treatment effect; ATE = CATE everywhere. A function should accept named arguments matching covariate names defined in the DGP and return a numeric vector of individual-level causal effects (CATE). See Details.
`propensity`	Numeric scalar, preset string, or function. Treatment assignment probability. A scalar (e.g. `0.5`) gives a constant propensity (randomized trial). Preset strings `"low"`, `"moderate"`, `"high"` generate a logistic propensity over confounders with coefficients 0.25, 0.5, and 1.0 respectively. A function follows the same

named-argument convention as `effect` and must return values in [0, 1]. Defaults to `"moderate"`.

`baseline` Numeric scalar, preset string, or function. Mean potential outcome under control, `E[Y(0) | W]`. Preset strings follow the same levels as `propensity` and apply a linear combination of confounders. Defaults to `"moderate"`, so that confounders declared via `n_confounders` (or `role = "confounder"`) enter both the treatment and outcome models and therefore actually induce confounding bias. With no confounders present, any preset baseline resolves to `0`. Set a numeric scalar (e.g. `0`) for a constant baseline that ignores covariates.

`sigma` Positive numeric. Standard deviation of the outcome noise term. Default `1`.

`covariates` Named list of `causalsim_covar()` objects (Option A / explicit path). Each name becomes the column name in generated data and the argument name expected by `effect`, `propensity`, and `baseline` functions. Merged with any auto-generated covariates; name collisions error.

`n_confounders` Non-negative integer. Standard normal confounders auto-generated as `W` (single) or `W1, W2, ...` (multiple).

`n_effect_modifiers` Non-negative integer. Auto-generates standard normal effect modifiers as `V` or `V1, V2, ...`.

`n_noise` Non-negative integer. Auto-generates standard normal noise covariates as `X` or `X1, X2, ...`.

`mc_draws` Positive integer. Monte Carlo draws for true ATE approximation. Default `10000L`. Ignored for scalar `effect`.

`seed` Integer or `NULL`. Passed to `set.seed()` before drawing. Default `NULL`.

Details

The returned data frame includes all covariate columns, the treatment indicator `A`, the outcome `Y`, and two ground-truth columns:

`.tau` Individual-level causal effect (CATE).

`.p` True propensity score.

For full control over the DGP (multiple draws, grid evaluation, estimator benchmarking), use `causalsim_dgp()` and `causalsim_draw()` directly.

Value

A data frame with covariate columns, `A`, `Y`, `.tau`, and `.p`.

Examples

```
# One confounder, constant effect of 2, moderate confounding
data <- causalsim(n = 500, n_confounders = 1, effect = 2, seed = 1L)
head(data)

# Heterogeneous effect with an explicit covariate spec
```

```
data2 <- causalsim(
  n = 500,
  covariates = list(
    W = causalsim_covar("normal", role = "confounder"),
    V = causalsim_covar("binary", role = "effect_modifier", prob = 0.4)
  ),
  effect = function(V) 2 + 1.5 * V,
  propensity = function(W) plogis(0.5 * W),
  seed = 42L
)
head(data2)
```

`causalsim_covar` *Define a covariate for a causal DGP*

Description

Constructs a fully-specified covariate descriptor for use in `causalsim_dgp()`. Every covariate has a distribution family, optional distribution parameters, and one or more causal roles that control how it enters the automatically-generated treatment and outcome models.

Usage

```
causalsim_covar(dist = "normal", role = "confounder", ...)
```

Arguments

`dist`	Character. Distribution family. One of `"normal"`, `"binary"`, `"uniform"`.
`role`	Character vector. Causal role(s). One or more of: `"confounder"` Enters both the propensity model and the outcome baseline; creates confounding bias in naive estimators. `"effect_modifier"` Available for use in the `effect` function; excluded from the propensity model unless also a `"confounder"`. `"noise"` Independent of both treatment and outcome; adds variance without confounding. Roles are not mutually exclusive: `role = c("confounder", "effect_modifier")` specifies a variable that both confounds and moderates the effect.
`...`	Distribution parameters. • `"normal"`: `mean` (default 0), `sd` (default 1) • `"binary"`: `prob` (default 0.5) • `"uniform"`: `min` (default 0), `max` (default 1)

Value

An S3 object of class `causalsim_covar`.

Examples

```
# Standard normal confounder
causalsim_covar("normal", role = "confounder", mean = 0, sd = 1)

# Binary effect modifier
causalsim_covar("binary", role = "effect_modifier", prob = 0.4)

# Variable that both confounds and moderates the effect
causalsim_covar("normal", role = c("confounder", "effect_modifier"))
```

`causalsim_dgp` *Create a causal data generating process*

Description

Defines a causal DGP with known ground truth. Covariates can be specified via shorthand count arguments (Option B), an explicit named list of `causalsim_covar()` objects (Option A), or both combined.

Usage

```
causalsim_dgp(
  n,
  effect = 1,
  propensity = "moderate",
  baseline = "moderate",
  sigma = 1,
  covariates = list(),
  n_confounders = 0L,
  n_effect_modifiers = 0L,
  n_noise = 0L,
  mc_draws = 10000L
)
```

Arguments

`n`	Positive integer. Sample size for each simulated dataset.
`effect`	Numeric scalar or function. A scalar specifies a constant (homogeneous) treatment effect; ATE = CATE everywhere. A function should accept named arguments matching covariate names defined in the DGP and return a numeric vector of individual-level causal effects (CATE). See Details.
`propensity`	Numeric scalar, preset string, or function. Treatment assignment probability. A scalar (e.g. `0.5`) gives a constant propensity (randomized trial). Preset strings `"low"`, `"moderate"`, `"high"` generate a logistic propensity over confounders with coefficients 0.25, 0.5, and 1.0 respectively. A function follows the same named-argument convention as `effect` and must return values in [0, 1]. Defaults to `"moderate"`.

`baseline` Numeric scalar, preset string, or function. Mean potential outcome under control, `E[Y(0) | W]`. Preset strings follow the same levels as `propensity` and apply a linear combination of confounders. Defaults to `"moderate"`, so that confounders declared via `n_confounders` (or `role = "confounder"`) enter both the treatment and outcome models and therefore actually induce confounding bias. With no confounders present, any preset baseline resolves to `0`. Set a numeric scalar (e.g. `0`) for a constant baseline that ignores covariates.

`sigma` Positive numeric. Standard deviation of the outcome noise term. Default `1`.

`covariates` Named list of `causalsim_covar()` objects (Option A / explicit path). Each name becomes the column name in generated data and the argument name expected by `effect`, `propensity`, and `baseline` functions. Merged with any auto-generated covariates; name collisions error.

`n_confounders` Non-negative integer. Standard normal confounders auto-generated as `W` (single) or `W1, W2, ...` (multiple).

`n_effect_modifiers` Non-negative integer. Auto-generates standard normal effect modifiers as `V` or `V1, V2, ...`.

`n_noise` Non-negative integer. Auto-generates standard normal noise covariates as `X` or `X1, X2, ...`.

`mc_draws` Positive integer. Monte Carlo draws for true ATE approximation. Default `10000L`. Ignored for scalar `effect`.

Details

Structural model:

```
W ~ covariate_spec
A ~ Bernoulli(propensity(W))
Y  = baseline(W) + effect(W) * A + N(0, sigma^2)
```

`A` is used for treatment throughout to avoid collision with R's built-in `T` alias.

Function calling convention:

The `effect`, `propensity`, and `baseline` functions are called with named arguments matching covariate names, not a data frame. Write:

```
effect = function(W) 2 + 1.5 * W
propensity = function(W1, W2) plogis(0.3 * W1 + 0.5 * W2)
```

Every argument name is validated against the DGP's covariate spec at construction time, so mismatches surface immediately rather than at draw time. The propensity function is additionally evaluated on a small test draw to confirm it returns values in [0, 1].

True ATE:

For scalar `effect` the true ATE is exact. For function `effect` it is approximated via Monte Carlo over `mc_draws` draws from the covariate distribution.

Value

An S3 object of class `causalsim_dgp` with components:

`n` Sample size (integer)

`covar_spec` Named list of `causalsim_covar()` objects

`effect_fn` Normalized effect function

`propensity_fn` Normalized propensity function

`baseline_fn` Normalized baseline function

`sigma` Outcome noise standard deviation

`true_ate` True ATE: exact for scalar, MC approximation otherwise

`heterogeneous` Logical; `TRUE` if `effect` was a function

`mc_draws` Monte Carlo draws used (integer)

Examples

```
# Minimal: one confounder, constant effect, moderate confounding
dgp <- causalsim_dgp(n = 500, n_confounders = 1, effect = 2)
dgp

# Heterogeneous effect, explicit covariate spec
dgp2 <- causalsim_dgp(
  n = 500,
  covariates = list(
    W = causalsim_covar("normal", role = "confounder"),
    V = causalsim_covar("binary", role = "effect_modifier", prob = 0.4)
  ),
  effect = function(V) 2 + 1.5 * V,
  propensity = function(W) plogis(0.5 * W),
  baseline = function(W) 1.5 * W
)

# Mixed: shorthand confounders + explicit noise covariate + RCT propensity
dgp3 <- causalsim_dgp(
  n = 1000,
  n_confounders = 2,
  covariates = list(X = causalsim_covar("normal", role = "noise")),
  effect = 1,
  propensity = 0.5
)
```

`causalsim_draw` *Draw a dataset from a causal DGP*

Description

Simulates one dataset from a `causalsim_dgp()` object using the structural model:

Usage

```
causalsim_draw(dgp, seed = NULL)
```

Arguments

dgp	A causalsim_dgp object created by causalsim_dgp().
seed	Integer or NULL. If non-null, passed to set.seed() before generating any random values, making the draw reproducible. Default NULL.

Details

```
W ~ covariate_spec
A ~ Bernoulli(propensity(W))
Y  = baseline(W) + effect(W) * A + N(0, sigma^2)
```

Value

A data frame with dgp$n rows and the following columns:

Covariate columns One column per covariate in dgp$covar_spec, named to match the spec (e.g. W, W1, Z).

A Binary treatment indicator (0/1).

Y Observed outcome.

.tau Individual treatment effect (CATE). Ground truth.

.p Individual propensity score. Ground truth.

The .tau and .p columns carry individual-level ground truth and are prefixed with . to distinguish them from observed variables. causalsim_eval() uses them directly without re-estimation.

Examples

```
dgp <- causalsim_dgp(n = 500, n_confounders = 1, effect = 2)
d   <- causalsim_draw(dgp, seed = 1)
head(d)

# Reproducible draws
d1 <- causalsim_draw(dgp, seed = 42)
d2 <- causalsim_draw(dgp, seed = 42)
identical(d1, d2)  # TRUE
```

`causalsim_eval` *Evaluate a causal estimator against a known DGP*

Description

Repeatedly draws datasets from a `causalsim_dgp()` object, applies a user-supplied estimator to each draw, and returns a tidy summary of estimator performance against the known ground truth.

Usage

```
causalsim_eval(
  dgp,
  estimator,
  reps = 200L,
  metrics = c("bias", "rmse", "coverage", "power"),
  seed = NULL
)
```

Arguments

`dgp`	A `causalsim_dgp` object.
`estimator`	A function that accepts a data frame produced by `causalsim_draw()` and returns a named numeric vector or single-row data frame. Must include an `estimate` field. For `"coverage"` and `"power"` metrics, must also include `ci_lower` and `ci_upper`. See Details.
`reps`	Positive integer. Number of simulation replications. Default `200L`.
`metrics`	Character vector. Subset of `"bias"`, `"rmse"`, `"coverage"`, `"power"`. Default: all four.
`seed`	Integer or `NULL`. Passed to `set.seed()` before the first replication. Default `NULL`.

Details

Estimator convention:

The estimator receives the full data frame returned by `causalsim_draw()`, including ground-truth columns `.tau` and `.p`. It should return a named numeric vector or one-row data frame with at minimum:

```
my_estimator <- function(data) {
  fit <- lm(Y ~ A + W, data = data)
  est <- coef(fit)["A"]
  se <- sqrt(vcov(fit)["A", "A"])
  c(estimate = est, ci_lower = est - 1.96 * se, ci_upper = est + 1.96 * se)
}
```

Metrics:

`"bias"` Monte Carlo estimate of bias: `mean(estimate) - true_ate`.

`"rmse"` Root mean squared error: `sqrt(mean((estimate - true_ate)^2))`.

`"coverage"` Proportion of replications where the CI brackets `true_ate`. Requires `ci_lower` and `ci_upper`.

`"power"` Proportion of replications where the CI excludes zero, i.e., the rejection rate of H0: ATE = 0. When `true_ate != 0`, this is power; when `true_ate == 0`, it is the Type I error rate. Requires `ci_lower` and `ci_upper`.

Monte Carlo standard errors (MCSE) are reported alongside each metric. For bias: `sd(estimates) / sqrt(reps)`. For RMSE: delta-method approximation. For coverage and power: Bernoulli SE.

Raw draws:

Per-replication estimates are stored in `result$draws` and can be used for custom analysis or plotting.

Value

An S3 object of class `causalsim_eval` with components:

`metrics` Tidy data frame: `metric`, `value`, `se`.

`draws` Data frame of per-replication estimates, one row per rep.

`true_ate` The DGP's true ATE.

`reps` Number of replications run.

Examples

```
dgp <- causalsim_dgp(n = 300, n_confounders = 1, effect = 2,
                     propensity = 0.5)

# OLS estimator (unbiased under RCT)
ols_estimator <- function(data) {
  fit <- lm(Y ~ A + W, data = data)
  est <- coef(fit)[["A"]]
  se <- sqrt(vcov(fit)["A", "A"])
  c(estimate = est, ci_lower = est - 1.96 * se, ci_upper = est + 1.96 * se)
}


result <- causalsim_eval(dgp, ols_estimator, reps = 100L, seed = 1L)
result
```

`causalsim_eval_grid` *Evaluate an estimator across a parameter grid*

Description

Runs `causalsim_eval()` over the Cartesian product of one or more DGP parameter values, returning a tidy data frame of performance metrics for every cell. Designed for studying how estimator behavior changes with sample size, confounding strength, effect size, or noise level.

Usage

```
causalsim_eval_grid(
  dgp,
  estimator,
  vary,
  reps = 200L,
  metrics = c("bias", "rmse", "coverage", "power"),
  seed = NULL,
  verbose = FALSE
)
```

Arguments

dgp	A causalsim_dgp object. Provides fixed parameter values for all dimensions not listed in vary.
estimator	A function as accepted by causalsim_eval().
vary	Named list of atomic vectors. Each name must be a valid causalsim_dgp() argument (except covariates, which cannot be varied atomically). Each vector supplies the values to try for that dimension. The full grid is the Cartesian product of all dimensions.
reps	Positive integer. Replications per grid cell. Default 200L.
metrics	Character vector. Passed to causalsim_eval().
seed	Integer or NULL. Passed to set.seed() before the first cell. The same RNG stream continues across cells, so results are jointly reproducible. Default NULL.
verbose	Logical. If TRUE, prints a progress message before each cell. Default FALSE.

Details

How it works:

For each cell in expand.grid(vary), causalsim_eval_grid() takes dgp's stored original parameters, overrides the cell's values, reconstructs a new causalsim_dgp(), runs causalsim_eval(), and tags the resulting metrics with the cell's parameter values.

Constraints on vary:

Elements of vary must be atomic vectors (character, numeric, integer). Functions cannot be varied via this interface. covariates (a list of causalsim_covar() objects) is excluded. For complex covariate variation, construct DGPs manually and use causalsim_eval() directly.

Note that varying n_confounders from 1 to 2 changes auto-generated covariate names from W to W1, W2. If the base DGP's effect function references W, the reconstructed DGP will error at construction time. Design the base DGP accordingly.

Value

An S3 object of class causalsim_eval_grid with components:

results Tidy data frame: one row per (cell, metric). Columns: grid parameter names, metric, value, se.

`grid` Data frame of grid cells, one row per cell.

`vary` Character vector of varied parameter names.

`reps` Replications per cell.

`metrics` Metrics evaluated.

Examples

```
dgp <- causalsim_dgp(n = 500, n_confounders = 1, effect = 2,
                     propensity = 0.5)

ols_estimator <- function(data) {
  fit <- lm(Y ~ A + W, data = data)
  est <- coef(fit)[["A"]]
  se  <- sqrt(vcov(fit)["A", "A"])
  c(estimate = est, ci_lower = est - 1.96 * se, ci_upper = est + 1.96 * se)
}


grid_result <- causalsim_eval_grid(
  dgp = dgp,
  estimator = ols_estimator,
  vary = list(n = c(250L, 500L, 1000L)),
  reps = 50L,
  seed = 1L
)
grid_result
```

`plot.causalsim_eval` *Plot the distribution of estimates from a causalsim_eval result*

Description

Draws a histogram of per-replication estimates with vertical lines marking the true ATE (solid) and the mean estimate (dashed).

Usage

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

Arguments

`x`	A `causalsim_eval` object.
`...`	Additional arguments passed to `graphics::hist()`.

Value

`x`, invisibly.

summary.causalsim_eval *Summarise a causalsim_eval result*

Description

Returns a structured summary of estimator performance including the metrics table and the distribution of per-replication estimates (mean, SD, median, and 10th / 90th percentiles).

Usage

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

Arguments

object	A causalsim_eval object.
...	Ignored.

Value

An S3 object of class causalsim_eval_summary with components:

metrics The metrics data frame from the original result.

mean_estimate Mean of per-replication estimates.

sd_estimate Standard deviation of per-replication estimates.

median_estimate Median of per-replication estimates.

p10_estimate 10th percentile of per-replication estimates.

p90_estimate 90th percentile of per-replication estimates.

true_ate True ATE from the DGP.

reps Number of replications.

Index