Package ‘propensity’

March 2, 2026

Title A Toolkit for Calculating and Working with Propensity Scores

Version 0.1.0

Description Calculates propensity score weights for multiple causal 'estimands' across binary, continuous, and categorical exposures. Provides methods for handling extreme propensity scores through trimming, truncation, and calibration. Includes inverse probability weighted estimators that correctly account for propensity score estimation uncertainty.

License MIT + file LICENSE

URL https://r-causal.github.io/propensity/, https://github.com/r-causal/propensity

BugReports https://github.com/r-causal/propensity/issues

Depends R (>= 4.2.0)

Imports cli, lifecycle, rlang, stats, tidyselect, vctrs (>= 0.6.5)

Suggests dplyr, ggplot2, knitr, mgcv, nnet, parsnip, probably, PSweight, rmarkdown, spelling, testthat (>= 3.0.0), tibble, tidyr, WeightIt, withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

NeedsCompilation no

Author Malcolm Barrett [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-0299-5825>)

Maintainer Malcolm Barrett <malcolmbarrett@gmail.com>

Repository CRAN

Date/Publication 2026-03-02 21:50:02 UTC

Contents

`propensity-package` *propensity: A Toolkit for Calculating and Working with Propensity Scores*

Description

propensity provides tools for propensity score analysis in causal inference. Calculate propensity score weights for a variety of causal estimands, handle extreme propensity scores through trimming, truncation, and calibration, and estimate causal effects with inverse probability weighting. The package supports binary, categorical, and continuous exposures.

Weight functions

Calculate propensity score weights for different causal estimands:

- `wt_ate()`: Average treatment effect (ATE) weights
- `wt_att()`: Average treatment effect on the treated (ATT) weights
- `wt_atu()`: Average treatment effect on the untreated (ATU) weights (`wt_atc()` is an alias)
- `wt_atm()`: Average treatment effect for the evenly matchable (ATM) weights
- `wt_ato()`: Average treatment effect for the overlap population (ATO) weights
- `wt_entropy()`: Entropy balancing weights
- `wt_cens()`: Censoring weights

Propensity score modifications

Handle extreme propensity scores before calculating weights:

- `ps_trim()`: Trim observations with extreme propensity scores
- `ps_trunc()`: Truncate (winsorize) extreme propensity scores
- `ps_calibrate()`: Calibrate propensity scores to improve balance
- `ps_refit()`: Re-estimate the propensity score model after trimming

Estimation

- `ipw()`: Inverse probability weighted estimator with variance estimation that accounts for propensity score estimation uncertainty

PSW class

The `psw()` class represents propensity score weights with metadata about the estimand and modifications applied:

- `psw()`, `as_psw()`, `is_psw()`: Create and test propensity score weights
- `estimand()`: Query the causal estimand
- `is_stabilized()`: Check if weights are stabilized

Author(s)

Maintainer: Malcolm Barrett `<malcolmbarrett@gmail.com>` (ORCID) [copyright holder]

See Also

- `vignette("propensity")` for a getting started guide
- The package website for full documentation

`ipw` *Inverse Probability Weighted Estimation*

Description

`ipw()` is a bring-your-own-model (BYOM) inverse probability weighted estimator for causal inference. You supply a fitted propensity score model and a fitted weighted outcome model; `ipw()` computes causal effect estimates with standard errors that correctly account for the two-step estimation process.

`ipw()` currently supports binary exposures with binary or continuous outcomes. For binary outcomes, it returns the risk difference, log risk ratio, and log odds ratio. For continuous outcomes, it returns the difference in means.

Usage

```
ipw(
  ps_mod,
  outcome_mod,
  .data = NULL,
  estimand = NULL,
  ps_link = NULL,
  conf_level = 0.95
)

## S3 method for class 'ipw'
as.data.frame(x, row.names = NULL, optional = NULL, exponentiate = FALSE, ...)
```

Arguments

ps_mod	A fitted propensity score model of class stats::glm(), typically a logistic regression with the exposure as the left-hand side of the formula.
outcome_mod	A fitted weighted outcome model of class stats::glm() or stats::lm(), with the outcome as the dependent variable and propensity score weights supplied via the weights argument. The weights should be created with a propensity weight function such as wt_ate().
.data	A data frame containing the exposure, outcome, and covariates. If NULL (the default), ipw() extracts data from the model objects. Supply .data explicitly if the outcome model formula contains transformations that prevent extraction of the exposure variable from stats::model.frame().
estimand	A character string specifying the causal estimand: "ate", "att", "ato", or "atm". If NULL, the estimand is inferred from the weights in outcome_mod. Auto-detection requires weights created with wt_ate(), wt_att(), wt_atm(), or wt_ato().
ps_link	A character string specifying the link function used in the propensity score model: "logit", "probit", or "cloglog". Defaults to the link used by ps_mod.
conf_level	Confidence level for intervals. Default is 0.95.
x	An ipw object.
row.names, optional, ...	Passed to base::as.data.frame().
exponentiate	If TRUE, exponentiate the log risk ratio and log odds ratio to produce risk ratios and odds ratios on their natural scale. The confidence interval bounds are also exponentiated. Standard errors, z statistics, and p-values remain on the log scale. Default is FALSE.

Value

An S3 object of class ipw with the following components:

estimand The causal estimand: one of "ate", "att", "ato", or "atm".

ps_mod The fitted propensity score model.

`outcome_mod` The fitted outcome model.

`estimates` A data frame with one row per effect measure and the following columns: `effect` (the measure name), `estimate` (point estimate), `std.err` (standard error), `z` (z-statistic), `ci.lower` and `ci.upper` (confidence interval bounds), `conf.level`, and `p.value`.

`as.data.frame()` returns the `estimates` component as a data frame. When `exponentiate = TRUE`, the `log(rr)` and `log(or)` rows are transformed: point estimates and confidence limits are exponentiated and the effect labels become `"rr"` and `"or"`.

Workflow

`ipw()` is designed around a three-step workflow:

1. Fit a propensity score model (e.g., logistic regression of exposure on confounders).
2. Calculate propensity score weights for your estimand (e.g., `wt_ate()`) and fit a weighted outcome model.
3. Pass both models to `ipw()` to obtain causal effect estimates with correct standard errors.

You are responsible for specifying and fitting both models. `ipw()` handles the variance estimation.

Effect measures

For binary outcomes (`stats::glm()` with `family = binomial()`), `ipw()` returns three effect measures:

- `rd`: Risk difference (marginal risk in exposed minus unexposed)
- `log(rr)`: Log risk ratio
- `log(or)`: Log odds ratio

For continuous outcomes (`stats::lm()` or `stats::glm()` with `family = gaussian()`), only the difference in means (`diff`) is returned.

Use `as.data.frame()` with `exponentiate = TRUE` to obtain risk ratios and odds ratios on their natural scale.

Variance estimation

Standard errors are computed via linearization, which correctly accounts for the uncertainty introduced by estimating propensity scores. This avoids the known problem of underestimated standard errors that arises from treating estimated weights as fixed. See Kostouraki et al. (2024) for details.

References

Kostouraki A, Hajage D, Rachet B, et al. On variance estimation of the inverse probability-of-treatment weighting estimator: A tutorial for different types of propensity score weights. *Statistics in Medicine*. 2024;43(13):2672–2694. doi:10.1002/sim.10078

See Also

`wt_ate()`, `wt_att()`, `wt_atm()`, `wt_ato()` for calculating propensity score weights.

`ps_trim()`, `ps_trunc()` for handling extreme propensity scores before weighting.

Examples

```
# Simulate data with a confounder, binary exposure, and binary outcome
set.seed(123)
n <- 200
x1 <- rnorm(n)
z <- rbinom(n, 1, plogis(0.5 * x1))
y <- rbinom(n, 1, plogis(-0.5 + 0.8 * z + 0.3 * x1))
dat <- data.frame(x1, z, y)

# Step 1: Fit a propensity score model
ps_mod <- glm(z ~ x1, data = dat, family = binomial())

# Step 2: Calculate ATE weights and fit a weighted outcome model
wts <- wt_ate(ps_mod)
outcome_mod <- glm(y ~ z, data = dat, family = binomial(), weights = wts)

# Step 3: Estimate causal effects with correct standard errors
result <- ipw(ps_mod, outcome_mod)
result

# Exponentiate log-RR and log-OR to get RR and OR
as.data.frame(result, exponentiate = TRUE)

# Continuous outcome example
y_cont <- 2 + 0.8 * z + 0.3 * x1 + rnorm(n)
dat$y_cont <- y_cont
outcome_cont <- lm(y_cont ~ z, data = dat, weights = wts)
ipw(ps_mod, outcome_cont)
```

---

is_ps_calibrated   *Check if propensity scores are calibrated*

Description

`is_ps_calibrated()` tests whether `x` is a calibrated propensity score object (class `ps_calib`) or a `psw` object derived from calibrated scores.

Usage

```
is_ps_calibrated(x)
```

Arguments

`x`	An object to test.

Value

A single `TRUE` or `FALSE`.

See Also

`ps_calibrate()` to calibrate propensity scores.

Examples

```
ps <- runif(100)
exposure <- rbinom(100, 1, ps)

is_ps_calibrated(ps)

calibrated <- ps_calibrate(ps, exposure, smooth = FALSE)
is_ps_calibrated(calibrated)
```

`is_ps_trimmed` *Test whether propensity scores have been trimmed*

Description

`is_ps_trimmed()` returns `TRUE` if `x` is a `ps_trim` object or a `psw` object created from trimmed propensity scores, and `FALSE` otherwise. This tests whether the *object* carries trimming information, not which individual units were trimmed; see `is_unit_trimmed()` for that.

Usage

```
is_ps_trimmed(x)
```

Arguments

`x`	An object to test.

Value

A logical scalar (`TRUE` or `FALSE`).

See Also

`ps_trim()` for trimming propensity scores, `is_unit_trimmed()` to identify which units were trimmed, `ps_trim_meta()` to retrieve full trimming metadata.

Examples

```
ps <- c(0.05, 0.3, 0.6, 0.95)
trimmed <- ps_trim(ps, method = "ps", lower = 0.1, upper = 0.9)

is_ps_trimmed(trimmed)
is_ps_trimmed(ps)
```

is_ps_truncated *Test whether propensity scores have been truncated*

Description

is_ps_truncated() returns TRUE if x is a ps_trunc object or a psw object derived from truncated propensity scores. Use is_unit_truncated() to find out *which* observations were modified.

Usage

```
is_ps_truncated(x)
```

Arguments

x	An object.

Value

A single TRUE or FALSE.

See Also

ps_trunc(), is_unit_truncated(), ps_trunc_meta()

Examples

```
ps <- c(0.02, 0.3, 0.5, 0.7, 0.98)
is_ps_truncated(ps)

ps_t <- ps_trunc(ps, method = "ps", lower = 0.05, upper = 0.95)
is_ps_truncated(ps_t)
```

is_refit *Check if propensity scores have been refit*

Description

is_refit() tests whether x is a ps_trim object whose propensity model has been refit on the retained (non-trimmed) observations via ps_refit().

Usage

```
is_refit(x)
```

Arguments

x   An object to test (typically a ps_trim vector).

Value

A single TRUE or FALSE.

See Also

ps_refit() to refit a propensity model after trimming, ps_trim() to trim propensity scores.

Examples

```
set.seed(2)
n <- 30
x <- rnorm(n)
z <- rbinom(n, 1, plogis(0.4 * x))
fit <- glm(z ~ x, family = binomial)
ps <- predict(fit, type = "response")

trimmed <- ps_trim(ps, lower = 0.2, upper = 0.8)
is_refit(trimmed)

refit <- ps_refit(trimmed, fit)
is_refit(refit)
```

---

is_unit_trimmed   *Identify which units were trimmed*

Description

is_unit_trimmed() returns a logical vector indicating which observations were removed by trimming. This is a per-unit query, as opposed to is_ps_trimmed(), which tests whether the object has been trimmed at all.

Usage

```
is_unit_trimmed(x)
```

Arguments

x   A ps_trim object created by ps_trim().

Value

A logical vector the same length as x, where TRUE marks a trimmed unit.

See Also

ps_trim() for trimming propensity scores, is_ps_trimmed() to test whether an object has been trimmed, ps_trim_meta() to retrieve full trimming metadata.

Examples

```
ps <- c(0.05, 0.3, 0.6, 0.95)
trimmed <- ps_trim(ps, method = "ps", lower = 0.1, upper = 0.9)

is_unit_trimmed(trimmed)

# Use to subset data to retained observations
kept <- !is_unit_trimmed(trimmed)
ps[kept]
```

---

is_unit_truncated *Identify which units were truncated*

Description

is_unit_truncated() returns a logical vector indicating which observations had their propensity scores modified by truncation. Use is_ps_truncated() to test whether an object has been truncated at all.

Usage

```
is_unit_truncated(x)
```

Arguments

x	A ps_trunc object created by ps_trunc().

Value

A logical vector the same length as x (or number of rows for matrix input). TRUE marks observations whose values were winsorized.

See Also

ps_trunc(), is_ps_truncated(), ps_trunc_meta()

Examples

```
ps <- c(0.02, 0.3, 0.5, 0.7, 0.98)
ps_t <- ps_trunc(ps, method = "ps", lower = 0.05, upper = 0.95)
is_unit_truncated(ps_t)
```

psw *Propensity Score Weight Vectors*

Description

psw objects are numeric vectors that carry metadata about propensity score weights, including the target estimand and whether the underlying propensity scores were trimmed, truncated, or calibrated.

Most users will encounter psw objects as return values from wt_ate() and related weight functions. These constructor and helper functions are useful for inspecting weight objects or for package developers extending propensity.

Usage

```
new_psw(
  x = double(),
  estimand = NULL,
  stabilized = FALSE,
  trimmed = FALSE,
  truncated = FALSE,
  calibrated = FALSE,
  ...
)

psw(
  x = double(),
  estimand = NULL,
  stabilized = FALSE,
  trimmed = FALSE,
  truncated = FALSE,
  calibrated = FALSE
)

is_psw(x)

is_stabilized(wt)

is_causal_wt(x)

as_psw(x, estimand = NULL)

estimand(wt)

estimand(wt) <- value
```

Arguments

`x`	For `psw()` and `new_psw()`: a numeric vector of weights (default: `double()`). For `is_psw()`, `is_causal_wt()`, and `as_psw()`: an object to test or coerce.
`estimand`	A character string identifying the target estimand (e.g., `"ate"`, `"att"`, `"ato"`). Defaults to `NULL`.
`stabilized`	Logical. Were the weights stabilized? Defaults to `FALSE`.
`trimmed`	Logical. Were the weights derived from trimmed propensity scores? Defaults to `FALSE`.
`truncated`	Logical. Were the weights derived from truncated propensity scores? Defaults to `FALSE`.
`calibrated`	Logical. Were the weights derived from calibrated propensity scores? Defaults to `FALSE`.
`...`	Additional attributes stored on the object (developer use only).
`wt`	A `psw` or `causal_wts` object.
`value`	A character string: the new estimand to assign.

Details

Constructors:

- `psw()` is the **user-facing** constructor. It coerces `x` to double and validates inputs before creating the object.
- `new_psw()` is the **low-level** constructor intended for developers. It assumes `x` is already a double vector and performs minimal validation.
- `as_psw()` coerces an existing numeric vector to a `psw` object.

Queries:

- `is_psw()` tests whether an object is a `psw` vector.
- `is_causal_wt()` tests whether an object inherits from the broader `causal_wts` class (which includes `psw` objects).
- `estimand()` and `estimand<-` get and set the estimand attribute.
- `is_stabilized()` returns `TRUE` if the weights are stabilized.

Arithmetic and combining:

Arithmetic operations on `psw` objects preserve the class and attributes, so operations like normalization (`weights / sum(weights)`) retain metadata. Combining `psw` objects with `c()` preserves the class only when all metadata matches; mismatched metadata produces a warning and falls back to a plain numeric vector.

Subsetting with `[` preserves class and attributes. Summary functions (`sum()`, `mean()`, etc.) return plain numeric values.

Value

- `new_psw()`, `psw()`, `as_psw()`: A `psw` vector.
- `is_psw()`, `is_causal_wt()`, `is_stabilized()`: A single logical value.
- `estimand()`: A character string, or `NULL` if no estimand is set.
- `estimand<-`: The modified `psw` object (called for its side effect).

See Also

wt_ate(), wt_att(), wt_atu(), wt_atm(), wt_ato() for calculating propensity score weights (which return psw objects).

ps_trim(), ps_trunc(), and ps_calibrate() for modifying propensity scores before weight calculation.

Examples

```
# Create psw objects directly
w <- psw(c(1.2, 0.8, 1.5), estimand = "ate")
w

# Query metadata
is_psw(w)
estimand(w)
is_stabilized(w)

# Coerce a plain numeric vector
as_psw(c(1.0, 2.0), estimand = "att")

# Arithmetic preserves the psw class
w / sum(w)

# Combining: compatible metadata is preserved
x <- psw(c(1.2, 0.8), estimand = "ate")
y <- psw(c(1.1, 0.9), estimand = "ate")
c(x, y)

# Combining: incompatible metadata warns and returns numeric
x <- psw(c(1.2, 0.8), estimand = "ate")
y <- psw(c(1.1, 0.9), estimand = "att")
c(x, y)
```

---

ps_calibrate *Calibrate propensity scores*

Description

ps_calibrate() adjusts estimated propensity scores so they better reflect true treatment probabilities. This can improve the accuracy of inverse probability weights derived from a misspecified propensity score model.

Usage

```
ps_calibrate(
  ps,
  .exposure,
```

```
  method = c("logistic", "isoreg"),
  smooth = TRUE,
  .focal_level = NULL,
  .reference_level = NULL,
  .treated = NULL,
  .untreated = NULL
)
```

Arguments

- `ps` — A numeric vector of propensity scores between 0 and 1. Must not already be calibrated.
- `.exposure` — A binary vector of observed treatment assignments, the same length as `ps`.
- `method` — Calibration method. One of:
  - `"logistic"` (Default) Logistic calibration, also called Platt scaling. Fits a logistic regression of `.exposure` on `ps`, yielding a smooth, parametric correction. Works well with small samples and when the bias in `ps` is approximately monotone.
  - `"isoreg"` Isotonic regression. Fits a non-parametric, monotone step function. More flexible than logistic calibration because it makes no distributional assumption, but needs larger samples for stable estimates.
- `smooth` — Logical. When `method = "logistic"`, controls the form of the calibration model. If `TRUE` (default), fits a GAM with a spline on `ps` via `mgcv::gam()`; if `FALSE`, fits a simple logistic regression via `stats::glm()`. Ignored when `method = "isoreg"`.
- `.focal_level` — The value of `.exposure` representing the focal group (typically the treated group). If `NULL` (default), coding is determined automatically.
- `.reference_level` — The value of `.exposure` representing the reference group (typically the control group). If `NULL` (default), coding is determined automatically.
- `.treated` — **[Deprecated]** Use `.focal_level` instead.
- `.untreated` — **[Deprecated]** Use `.reference_level` instead.

Details

Calibration is useful when the propensity score model is correctly specified in terms of variable selection but produces probabilities that are systematically too high or too low. Unlike `ps_trim()` and `ps_trunc()`, which handle extreme scores by removing or bounding them, calibration reshapes the entire distribution of scores.

Choosing a method:

- Use `"logistic"` (the default) as a first choice. It is stable and fast, and the `smooth = TRUE` option adds flexibility via a spline.
- Use `"isoreg"` when you suspect a non-smooth or irregular relationship between estimated and true probabilities and have a sufficiently large sample.

The calibrated scores are returned as a `ps_calib` object, which can be passed directly to weight functions such as `wt_ate()`.

Value

A ps_calib vector the same length as ps. The attribute ps_calib_meta stores calibration metadata (method and whether smoothing was applied). Use is_ps_calibrated() to test whether an object has been calibrated.

References

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 61–74.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 694–699. doi:10.1145/775047.775151

See Also

is_ps_calibrated() to test for calibrated scores; ps_trim() and ps_trunc() for alternative approaches to extreme propensity scores; wt_ate() and other weight functions that accept ps_calib objects.

Examples

```
# Simulate data
set.seed(42)
ps <- runif(200)
exposure <- rbinom(200, 1, ps)

# Logistic calibration without smoothing (simple Platt scaling)
cal <- ps_calibrate(ps, exposure, smooth = FALSE)
cal

# Use calibrated scores to calculate weights
wt_ate(cal, exposure)

# Isotonic regression calibration
cal_iso <- ps_calibrate(ps, exposure, method = "isoreg")

if (rlang::is_installed("mgcv")) {
  # Logistic calibration with spline smoothing (default)
  cal_smooth <- ps_calibrate(ps, exposure)
}
```

---

ps_refit — *Refit a Propensity Score Model on Retained Observations*

Description

Re-estimates a propensity score model using only the observations retained after trimming. This is the recommended intermediate step between `ps_trim()` and weight calculation (e.g. `wt_ate()`):

`ps_trim()` **->** `ps_refit()` **->** `wt_*()`

Trimming changes the target population by removing observations with extreme propensity scores. Refitting the model on the retained subset produces propensity scores that better reflect this population, improving both model fit and downstream weight estimation. Weight functions warn if a trimmed propensity score has not been refit.

Usage

```
ps_refit(trimmed_ps, model, .data = NULL, ...)
```

Arguments

`trimmed_ps`	A `ps_trim` object returned by `ps_trim()`.
`model`	The original fitted model used to estimate the propensity scores (e.g. a glm or multinom object). The model is refit via update() on the retained subset.
`.data`	A data frame. If `NULL` (the default), the data are extracted from `model` via model.frame().
`...`	Additional arguments passed to update().

Value

A `ps_trim` object with re-estimated propensity scores for retained observations and `NA` for trimmed observations. Use `is_refit()` to confirm refitting was applied.

See Also

`ps_trim()` for the trimming step, `is_refit()` to check refit status, `wt_ate()` and other weight functions for the next step in the pipeline.

Examples

```
set.seed(2)
n <- 200
x <- rnorm(n)
z <- rbinom(n, 1, plogis(0.4 * x))

# fit a propensity score model
ps_model <- glm(z ~ x, family = binomial)
ps <- predict(ps_model, type = "response")

# trim -> refit -> weight pipeline
trimmed <- ps_trim(ps, lower = 0.1, upper = 0.9)
refit <- ps_refit(trimmed, ps_model)
wts <- wt_ate(refit, .exposure = z)

is_refit(refit)
```

ps_trim *Trim Propensity Scores*

Description

Trim observations with extreme propensity scores by replacing them with NA, effectively removing those units from downstream analyses. The returned object has the same length (or dimensions) as the input, with trimmed entries set to NA. After trimming, refit the propensity score model on the retained observations with ps_refit().

Usage

```
ps_trim(
  ps,
  method = c("ps", "adaptive", "pctl", "pref", "cr", "optimal"),
  lower = NULL,
  upper = NULL,
  .exposure = NULL,
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)
```

Arguments

ps
: A numeric vector of propensity scores in (0, 1) for binary exposures, or a matrix / data frame where each column gives the propensity score for one level of a categorical exposure.

method
: Trimming method. One of:

- "ps" (default): Fixed threshold. Observations with propensity scores outside [lower, upper] are trimmed. For categorical exposures, observations where *any* column falls below lower (the symmetric threshold delta) are trimmed.
- "adaptive": Data-driven threshold that minimizes the asymptotic variance of the IPW estimator (Crump et al., 2009). The lower and upper arguments are ignored.
- "pctl": Quantile-based. Observations outside the [lower, upper] quantiles of the propensity score distribution are trimmed. Defaults: lower = 0.05, upper = 0.95.
- "pref": Preference score trimming. Transforms propensity scores to the preference scale (Walker et al., 2013) and trims outside [lower, upper]. Requires .exposure. Binary exposures only. Defaults: lower = 0.3, upper = 0.7.

- "cr": Common range (clinical equipoise). Trims to the overlap region of the propensity score distributions across exposure groups. Requires .exposure. Binary exposures only. The lower and upper arguments are ignored.
- "optimal": Multi-category optimal trimming (Yang et al., 2016). Categorical exposures only. Requires .exposure.

For categorical exposures, only "ps" and "optimal" are supported.

lower, upper
: Numeric thresholds whose interpretation depends on method:
- "ps": absolute propensity score bounds (defaults: 0.1, 0.9). For categorical exposures, only lower is used as the symmetric threshold.
- "pctl": quantile probabilities (defaults: 0.05, 0.95).
- "pref": preference score bounds (defaults: 0.3, 0.7).
- "adaptive", "cr", "optimal": ignored (thresholds are data-driven).

.exposure
: An exposure variable. Required for "pref", "cr" (binary vector), and "optimal" (factor or character). Not required for other methods.

.focal_level
: The value of .exposure representing the focal (treated) group. For binary exposures, defaults to the higher value. Required for wt_att() and wt_atu() with categorical exposures.

.reference_level
: The value of .exposure representing the reference (control) group. Automatically detected if not supplied.

...
: Additional arguments passed to methods.

.treated
: **[Deprecated]** Use .focal_level instead.

.untreated
: **[Deprecated]** Use .reference_level instead.

Details

How trimming works:

Trimming identifies observations with extreme (near 0 or 1) propensity scores and sets them to NA. These observations are excluded from subsequent weight calculations and effect estimation. The goal is to remove units that lack sufficient overlap between exposure groups, which would otherwise receive extreme weights and destabilize estimates.

Choosing a method:

- Use "ps" when you have a specific threshold in mind or want a simple default.
- Use "adaptive" for a principled, data-driven cutoff that targets variance reduction.
- Use "pctl" to trim a fixed percentage of extreme values from each tail.
- Use "pref" when you want to restrict to the region of clinical equipoise based on the preference score.
- Use "cr" to restrict to the common support region where both exposure groups have observed propensity scores.
- Use "optimal" for multi-category (3+) exposures; this is the only data-driven method available for categorical treatments.

Typical workflow:

1. Fit a propensity score model
2. Apply `ps_trim()` to flag extreme values
3. Call `ps_refit()` to re-estimate propensity scores on the retained sample
4. Compute weights with `wt_ate()` or another weight function

Object behavior:

Arithmetic operations on `ps_trim` objects return plain numeric vectors, since transformed propensity scores (e.g., `1/ps`) are no longer propensity scores. Trimmed values propagate as `NA` in calculations; use `na.rm = TRUE` where appropriate.

When combining `ps_trim` objects with `c()`, trimming parameters must match. Mismatched parameters trigger a warning and return a numeric vector.

Use `ps_trim_meta()` to inspect the trimming metadata, including the method, cutoffs, and which observations were retained or trimmed.

Value

A `ps_trim` object (a numeric vector with class `"ps_trim"`, or a matrix with class `"ps_trim_matrix"`). Trimmed observations are `NA`. Metadata is stored in the `"ps_trim_meta"` attribute and can be accessed with `ps_trim_meta()`. Key fields include:

- `method`: the trimming method used
- `keep_idx`: integer indices of retained observations
- `trimmed_idx`: integer indices of trimmed (NA) observations
- Method-specific fields such as `cutoff` (adaptive), `q_lower`/`q_upper` (pctl), `cr_lower`/`cr_upper` (cr), `delta` (categorical ps), or `lambda` (optimal)

References

Crump, R. K., Hotz, V. J., Imbens, G. W., & Mitnik, O. A. (2009). Dealing with limited overlap in estimation of average treatment effects. *Biometrika*, 96(1), 187–199.

Walker, A. M., Patrick, A. R., Lauer, M. S., et al. (2013). A tool for assessing the feasibility of comparative effectiveness research. *Comparative Effectiveness Research*, 3, 11–20.

Yang, S., Imbens, G. W., Cui, Z., Faries, D. E., & Kadziola, Z. (2016). Propensity score matching and subclassification in observational studies with multi-level treatments. *Biometrics*, 72(4), 1055–1065.

See Also

`ps_trunc()` for bounding (winsorizing) instead of discarding, `ps_refit()` to re-estimate propensity scores after trimming, `ps_calibrate()` for calibration-based adjustment, `ps_trim_meta()` to inspect trimming metadata, `is_ps_trimmed()` and `is_unit_trimmed()` for logical queries.

Examples

```
set.seed(2)
n <- 300
x <- rnorm(n)
z <- rbinom(n, 1, plogis(1.3 * x))
```

```
fit <- glm(z ~ x, family = binomial)
ps <- predict(fit, type = "response")

# Fixed threshold trimming (default)
trimmed <- ps_trim(ps, method = "ps", lower = 0.1, upper = 0.9)
trimmed

# How many observations were trimmed?
sum(is_unit_trimmed(trimmed))

# Data-driven adaptive trimming
ps_trim(ps, method = "adaptive")

# Quantile-based trimming at 5th and 95th percentiles
ps_trim(ps, method = "pctl")

# Refit after trimming, then compute weights
trimmed <- ps_trim(ps, method = "adaptive")
refitted <- ps_refit(trimmed, fit)
wt_ate(refitted, .exposure = z)
```

`ps_trim_meta` *Extract trimming metadata from a* `ps_trim` *object*

Description

`ps_trim_meta()` returns the metadata list attached to a `ps_trim` object by `ps_trim()`.

Usage

```
ps_trim_meta(x)
```

Arguments

`x`	A `ps_trim` object.

Value

A named list with elements:

- `method` Character string indicating the trimming method used.
- `keep_idx` Integer vector of retained observation indices.
- `trimmed_idx` Integer vector of trimmed observation indices.
- `lower`, `upper` Numeric cutoffs, when applicable.
- `refit` Logical, `TRUE` if the model was refit via `ps_refit()`.

Additional method-specific elements (e.g. `cutoff`, `delta`, `lambda`) may also be present.

See Also

ps_trim() for trimming propensity scores, is_ps_trimmed() and is_unit_trimmed() for predicate queries.

Examples

```
ps <- c(0.05, 0.3, 0.6, 0.95)
trimmed <- ps_trim(ps, method = "ps", lower = 0.1, upper = 0.9)

ps_trim_meta(trimmed)
```

---

ps_trunc — *Truncate (Winsorize) Propensity Scores*

Description

ps_trunc() bounds extreme propensity scores to fixed limits, replacing out-of-range values with the boundary value (a form of *winsorizing*). The result is a vector or matrix of the same length and dimensions as ps, with no observations removed. This contrasts with ps_trim(), which sets extreme values to NA (effectively removing those observations from analysis).

Usage

```
ps_trunc(
  ps,
  method = c("ps", "pctl", "cr"),
  lower = NULL,
  upper = NULL,
  .exposure = NULL,
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)
```

Arguments

ps — A numeric vector of propensity scores between 0 and 1 (binary exposures), or a matrix/data.frame where each column contains propensity scores for one level of a categorical exposure.

method — One of "ps", "pctl", or "cr":

- "ps" (default): Truncate directly on propensity score values. Values outside [lower, upper] are set to the nearest bound. For categorical exposures, applies symmetric truncation using lower as the threshold (delta) and renormalizes rows to sum to 1.

- "pctl": Truncate at quantiles of the propensity score distribution. The lower and upper arguments specify quantile probabilities. For categorical exposures, quantiles are computed across all columns.
- "cr": Truncate to the common range of propensity scores across exposure groups (binary exposures only). Bounds are [min(ps[focal]), max(ps[reference])]. Requires .exposure.

lower, upper
: Bounds for truncation. Interpretation depends on method:
  - method = "ps": Propensity score values (defaults: 0.1 and 0.9). For categorical exposures, lower is the truncation threshold delta (default: 0.01); upper is ignored.
  - method = "pctl": Quantile probabilities (defaults: 0.05 and 0.95; categorical defaults: 0.01 and 0.99).
  - method = "cr": Ignored; bounds are determined by the data.

.exposure
: An exposure vector. Required for method "cr" (binary exposure vector) and for categorical exposures (factor or character vector) with any method.

.focal_level
: The value of .exposure representing the focal (treated) group. For binary exposures, defaults to the higher value. Required for wt_att() and wt_atu() with categorical exposures.

.reference_level
: The value of .exposure representing the reference (control) group. Automatically detected if not supplied.

...
: Additional arguments passed to methods.

.treated
: **[Deprecated]** Use .focal_level instead.

.untreated
: **[Deprecated]** Use .reference_level instead.

Details

Unlike ps_trim(), truncation preserves all observations. No NA values are introduced; out-of-range scores are replaced with the boundary value.

For **binary exposures**, each propensity score e_i is bounded:

- If $e_i < l$, set $e_i = l$ (the lower bound).
- If $e_i > u$, set $e_i = u$ (the upper bound).

For **categorical exposures**, values below the threshold are set to the threshold and each row is renormalized to sum to 1.

Arithmetic behavior: Arithmetic operations on ps_trunc objects return plain numeric vectors. Once propensity scores are transformed (e.g., into weights), the result is no longer a propensity score.

Combining behavior: Combining ps_trunc objects with c() requires matching truncation parameters. Mismatched parameters produce a warning and return a plain numeric vector.

Value

A ps_trunc object (a numeric vector for binary exposures, or a matrix for categorical exposures). Use ps_trunc_meta() to inspect metadata including method, lower_bound, upper_bound, and truncated_idx (positions of modified values).

References

Crump, R. K., Hotz, V. J., Imbens, G. W., & Mitnik, O. A. (2009). Dealing with limited overlap in estimation of average treatment effects. *Biometrika*, 96(1), 187–199.

Walker, A. M., Patrick, A. R., Lauer, M. S., et al. (2013). A tool for assessing the feasibility of comparative effectiveness research. *Comparative Effectiveness Research*, 3, 11–20.

See Also

`ps_trim()` for removing (rather than bounding) extreme values, `ps_refit()` for refitting the propensity model after trimming, `is_ps_truncated()`, `is_unit_truncated()`, `ps_trunc_meta()`

Examples

```
set.seed(2)
n <- 200
x <- rnorm(n)
z <- rbinom(n, 1, plogis(1.2 * x))
fit <- glm(z ~ x, family = binomial)
ps <- predict(fit, type = "response")

# Truncate to [0.1, 0.9]
ps_t <- ps_trunc(ps, method = "ps", lower = 0.1, upper = 0.9)
ps_t

# Truncate at the 1st and 99th percentiles
ps_trunc(ps, method = "pctl", lower = 0.01, upper = 0.99)

# Use truncated scores to calculate weights
wt_ate(ps_t, .exposure = z)

# Inspect which observations were truncated
is_unit_truncated(ps_t)
```

`ps_trunc_meta` *Extract truncation metadata from a* `ps_trunc` *object*

Description

Returns the metadata list attached to a `ps_trunc` object. The list includes fields such as `method`, `lower_bound`, `upper_bound`, and `truncated_idx`.

Usage

```
ps_trunc_meta(x)
```

Arguments

`x`	A `ps_trunc` object created by `ps_trunc()`.

Value

A named list with truncation metadata, including:

- `method` – the truncation method used (`"ps"`, `"pctl"`, or `"cr"`)
- `lower_bound`, `upper_bound` – the applied bounds
- `truncated_idx` – integer positions of values that were winsorized

See Also

`ps_trunc()`, `is_ps_truncated()`, `is_unit_truncated()`

Examples

```
ps <- c(0.02, 0.3, 0.5, 0.7, 0.98)
ps_t <- ps_trunc(ps, method = "ps", lower = 0.05, upper = 0.95)
ps_trunc_meta(ps_t)
```

---

wt_ate   *Calculate propensity score weights*

Description

Compute inverse probability weights for causal inference under different estimands. Each function targets a different population:

- `wt_ate()`: **Average Treatment Effect** – the full population.
- `wt_att()`: **Average Treatment Effect on the Treated** – the treated (focal) group.
- `wt_atu()`: **Average Treatment Effect on the Untreated** – the untreated (reference) group. `wt_atc()` is an alias.
- `wt_atm()`: **Average Treatment Effect for the Evenly Matchable** – units with the most overlap.
- `wt_ato()`: **Average Treatment Effect for the Overlap Population** – weights proportional to overlap.
- `wt_entropy()`: **Entropy-weighted Average Treatment Effect** – an entropy-balanced population.
- `wt_cens()`: **Inverse probability of censoring weights** – uses the same formula as `wt_ate()` but labels the estimand `"uncensored"`. Use these to adjust for censoring in survival analysis, not for treatment weighting.

`.propensity` accepts a numeric vector of predicted probabilities, a `data.frame` of per-level probabilities, a fitted `glm` object, or a modified propensity score created by `ps_trim()`, `ps_trunc()`, `ps_refit()`, or `ps_calibrate()`.

All functions return a `psw` object – a numeric vector that tracks the estimand, stabilization status, and any trimming or truncation applied.

Usage

```
wt_ate(
  .propensity,
  .exposure,
  .sigma = NULL,
  exposure_type = c("auto", "binary", "categorical", "continuous"),
  .focal_level = NULL,
  .reference_level = NULL,
  stabilize = FALSE,
  stabilization_score = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

## S3 method for class 'data.frame'
wt_ate(
  .propensity,
  .exposure,
  .sigma = NULL,
  exposure_type = c("auto", "binary", "categorical", "continuous"),
  .focal_level = NULL,
  .reference_level = NULL,
  stabilize = FALSE,
  stabilization_score = NULL,
  ...,
  .propensity_col = NULL,
  .treated = NULL,
  .untreated = NULL
)

wt_att(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

## S3 method for class 'data.frame'
wt_att(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
```

```
  .reference_level = NULL,
  ...,
  .propensity_col = NULL,
  .treated = NULL,
  .untreated = NULL
)

wt_atu(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

## S3 method for class 'data.frame'
wt_atu(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .propensity_col = NULL,
  .treated = NULL,
  .untreated = NULL
)

wt_atm(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

## S3 method for class 'data.frame'
wt_atm(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
```

```
  .reference_level = NULL,
  ...,
  .propensity_col = NULL,
  .treated = NULL,
  .untreated = NULL
)

wt_ato(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

## S3 method for class 'data.frame'
wt_ato(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .propensity_col = NULL,
  .treated = NULL,
  .untreated = NULL
)

wt_entropy(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

## S3 method for class 'data.frame'
wt_entropy(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
```

```
  .reference_level = NULL,
  ...,
  .propensity_col = NULL,
  .treated = NULL,
  .untreated = NULL
)

wt_atc(
  .propensity,
  .exposure,
  exposure_type = c("auto", "binary", "categorical"),
  .focal_level = NULL,
  .reference_level = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

wt_cens(
  .propensity,
  .exposure,
  .sigma = NULL,
  exposure_type = c("auto", "binary", "categorical", "continuous"),
  .focal_level = NULL,
  .reference_level = NULL,
  stabilize = FALSE,
  stabilization_score = NULL,
  ...,
  .treated = NULL,
  .untreated = NULL
)

## S3 method for class 'data.frame'
wt_cens(
  .propensity,
  .exposure,
  .sigma = NULL,
  exposure_type = c("auto", "binary", "categorical", "continuous"),
  .focal_level = NULL,
  .reference_level = NULL,
  stabilize = FALSE,
  stabilization_score = NULL,
  ...,
  .propensity_col = NULL,
  .treated = NULL,
  .untreated = NULL
)
```

Arguments

`.propensity` Propensity scores in one of several forms:

- A **numeric vector** of predicted probabilities (binary/continuous).
- A **data frame** or matrix with one column per exposure level (categorical), or two columns for binary (see `.propensity_col`).
- A fitted `glm` object – fitted values are extracted automatically.
- A modified propensity score created by `ps_trim()`, `ps_trunc()`, `ps_refit()`, or `ps_calibrate()`.

`.exposure` The exposure (treatment) variable. For binary exposures, a numeric 0/1 vector, logical, or two-level factor. For categorical exposures, a factor or character vector. For continuous exposures, a numeric vector. Optional when `.propensity` is a `glm` object (extracted from the model).

`.sigma` Numeric vector of observation-level standard deviations for continuous exposures (e.g., `influence(model)$sigma`). Extracted automatically when `.propensity` is a `glm` object.

`exposure_type` Type of exposure: `"auto"` (default), `"binary"`, `"categorical"`, or `"continuous"`. `"auto"` detects the type from `.exposure`.

`.focal_level` The value of `.exposure` representing the focal (treated) group. For binary exposures, defaults to the higher value. Required for `wt_att()` and `wt_atu()` with categorical exposures.

`.reference_level` The value of `.exposure` representing the reference (control) group. Automatically detected if not supplied.

`stabilize` If `TRUE`, multiply weights by an estimate of the marginal treatment probability (binary) or density (continuous). Only supported by `wt_ate()` and `wt_cens()`. See **Stabilization** in Details.

`stabilization_score` Optional numeric value to use as the stabilization multiplier instead of the default (the marginal mean of `.exposure`). Ignored when `stabilize = FALSE`.

`...` These dots are for future extensions and must be empty.

`.treated` **[Deprecated]** Use `.focal_level` instead.

`.untreated` **[Deprecated]** Use `.reference_level` instead.

`.propensity_col` Column to use when `.propensity` is a data frame with a binary exposure. Accepts a column name (quoted or unquoted) or numeric index. Defaults to the second column. Ignored for categorical exposures, where all columns are used.

Details

Exposure types:

All weight functions support binary exposures. `wt_ate()` and `wt_cens()` also support continuous exposures. All except `wt_cens()` support categorical exposures.

- **Binary**: `.exposure` is a two-level vector (e.g., 0/1, logical, or a two-level factor). `.propensity` is a numeric vector of P(treatment | X).

- **Categorical**: `.exposure` is a factor or character vector with 3+ levels. `.propensity` must be a matrix or data frame with one column per level, where rows sum to 1.
- **Continuous**: `.exposure` is a numeric vector. `.propensity` is a vector of conditional means (fitted values). Weights use a normal density ratio; stabilization is strongly recommended.
- **Auto** (default): Detects the exposure type from `.exposure`.

Stabilization:

Setting `stabilize = TRUE` multiplies the base weight by an estimate of P(A) (binary) or f_A(A) (continuous), reducing variance. When no `stabilization_score` is supplied, the marginal mean of `.exposure` is used. Stabilization is supported for ATE and censoring weights (`wt_ate()` and `wt_cens()`) and is strongly recommended for continuous exposures.

Handling extreme weights:

Extreme weights signal positivity violations, poor model fit, or limited overlap. You can address them by:

- Choosing an overlap-focused estimand (`wt_ato()`, `wt_atm()`, `wt_entropy()`), which down-weight units in regions of poor overlap.
- Trimming (`ps_trim()`) or truncating (`ps_trunc()`) propensity scores before computing weights.
- Calibrating weights with `ps_calibrate()`.
- Stabilizing ATE weights (`stabilize = TRUE`).

See the halfmoon package for weight diagnostics and visualization.

Weight formulas:

Binary exposures:

For binary treatments ($A \in \{0, 1\}$), with propensity score $e(X) = P(A = 1 \mid X)$:

- **ATE**: $w = \frac{A}{e(X)} + \frac{1-A}{1-e(X)}$
- **ATT**: $w = A + \frac{(1-A) \cdot e(X)}{1-e(X)}$
- **ATU**: $w = \frac{A \cdot (1-e(X))}{e(X)} + (1 - A)$
- **ATM**: $w = \frac{\min(e(X), 1-e(X))}{A \cdot e(X) + (1-A) \cdot (1-e(X))}$
- **ATO**: $w = A \cdot (1 - e(X)) + (1 - A) \cdot e(X)$
- **Entropy**: $w = \frac{h(e(X))}{A \cdot e(X) + (1-A) \cdot (1-e(X))}$, where $h(e) = -[e \log(e) + (1 - e) \log(1 - e)]$

Continuous exposures:

Weights use the density ratio $w = f_A(A)/f_{A|X}(A \mid X)$, where f_A is the marginal density and $f_{A|X}$ is the conditional density (both assumed normal). Only `wt_ate()` and `wt_cens()` support continuous exposures.

Categorical exposures:

For K-level treatments, weights take the tilting-function form $w_i = h(\mathbf{e}_i)/e_{i,Z_i}$, where e_{i,Z_i} is the propensity for unit i's observed level and $h(\cdot)$ depends on the estimand:

- **ATE**: $h(\mathbf{e}) = 1$
- **ATT**: $h(\mathbf{e}) = e_{\text{focal}}$
- **ATU**: $h(\mathbf{e}) = 1 - e_{\text{focal}}$
- **ATM**: $h(\mathbf{e}) = \min(e_1, \ldots, e_K)$
- **ATO**: $h(\mathbf{e}) = \left(\sum_k 1/e_k\right)^{-1}$
- **Entropy**: $h(\mathbf{e}) = -\sum_k e_k \log(e_k)$

Value

A psw vector (a double vector with class psw) carrying these attributes:

- estimand: character, e.g. "ate", "att", "uncensored".
- stabilized: logical, whether stabilization was applied.
- trimmed: logical, whether the propensity scores were trimmed.
- truncated: logical, whether the propensity scores were truncated.
- calibrated: logical, whether the propensity scores were calibrated.

References

Barrett, M., D'Agostino McGowan, L., & Gerke, T. *Causal Inference in R*. https://www.r-causal.org/

Rosenbaum, P. R., & Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1), 41–55.

Li, L., & Greene, T. (2013). A weighting analogue to pair matching in propensity score analysis. *The International Journal of Biostatistics*, 9(2), 215–234. (ATM weights)

Li, F., Morgan, K. L., & Zaslavsky, A. M. (2018). Balancing covariates via propensity score weighting. *Journal of the American Statistical Association*, 113(521), 390–400. (ATO weights)

Zhou, Y., Matsouaka, R. A., & Thomas, L. (2020). Propensity score weighting under limited overlap and model misspecification. *Statistical Methods in Medical Research*, 29(12), 3721–3756. (Entropy weights)

Hirano, K., & Imbens, G. W. (2004). The propensity score with continuous treatments. In *Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives* (pp. 73–84).

Austin, P. C., & Stuart, E. A. (2015). Moving towards best practice when using inverse probability of treatment weighting (IPTW). *Statistics in Medicine*, 34(28), 3661–3679.

See Also

- psw() for the returned weight vector class.
- ps_trim(), ps_trunc(), ps_refit(), and ps_calibrate() for modifying propensity scores before weighting.
- ipw() for inverse-probability-weighted estimation of causal effects.

Examples

```
# -- Binary exposure, numeric propensity scores ----------------------
set.seed(123)
ps <- runif(100, 0.1, 0.9)
trt <- rbinom(100, 1, ps)

wt_ate(ps, trt)
wt_att(ps, trt)
wt_atu(ps, trt)
wt_atm(ps, trt)
wt_ato(ps, trt)
```

```
wt_entropy(ps, trt)

# Stabilized ATE weights (reduces variance)
wt_ate(ps, trt, stabilize = TRUE)

# Inspect the result
w <- wt_ate(ps, trt)
estimand(w)
summary(w)

# -- Overlap-focused estimands handle extreme PS better --------------
ps_extreme <- c(0.01, 0.02, 0.98, 0.99, rep(0.5, 4))
trt_extreme <- c(0, 0, 1, 1, 0, 1, 0, 1)

max(wt_ate(ps_extreme, trt_extreme))
max(wt_ato(ps_extreme, trt_extreme))

# -- From a fitted GLM -----------------------------------------------
x1 <- rnorm(100)
x2 <- rnorm(100)
trt2 <- rbinom(100, 1, plogis(0.5 * x1 + 0.3 * x2))
ps_model <- glm(trt2 ~ x1 + x2, family = binomial)

# Exposure is extracted from the model automatically
wt_ate(ps_model)

# -- Data frame input ------------------------------------------------
ps_df <- data.frame(
  control = c(0.9, 0.7, 0.3, 0.1),
  treated = c(0.1, 0.3, 0.7, 0.9)
)
exposure <- c(0, 0, 1, 1)
wt_ate(ps_df, exposure)
wt_ate(ps_df, exposure, .propensity_col = "treated")

# -- Censoring weights -----------------------------------------------
cens_ps <- runif(50, 0.6, 0.95)
cens_ind <- rbinom(50, 1, cens_ps)
wt_cens(cens_ps, cens_ind)
estimand(wt_cens(cens_ps, cens_ind))  # "uncensored"
```

Index