

Package: positively (via r-universe)

October 4, 2026

Title Diagnose Positivity Violations and Extrapolation in Causal Inference

Version 0.0.0.9000

Description Provides diagnostics for positivity violations and extrapolation in causal inference. Includes methods based on effective data points, positivity regression trees, hat values, highest density regions, density ratios, and parametric bootstrap bias estimates, covering binary, categorical, and continuous exposures as well as time-varying treatments. Each diagnostic returns a tidy result object and an optional plot.

License MIT + file LICENSE

URL <https://r-causal.github.io/positively/>,
<https://github.com/r-causal/positively>

BugReports <https://github.com/r-causal/positively/issues>

Depends R (>= 4.3.0)

Imports causalgenerics (>= 0.1.0.9000), cli, generics, ggplot2 (>= 3.5.0), purrr, rlang, rpart, S7, stats, tibble, tidyselect, utils, vctrs

Suggests dplyr (>= 1.1.0), ggridges, halfmoon, knitr, lmt, lpSolve, nnet, patchwork, propensity (>= 0.1.0.9000), rmarkdown, spelling, testthat (>= 3.0.0), vdiff, withr

VignetteBuilder knitr

Remotes r-causal/causalgenerics, r-causal/propensity

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

Repository <https://r-causal.r-universe.dev>

Date/Publication 2026-10-04 21:01:13 UTC
RemoteUrl https://github.com/r-causal/positively
RemoteRef HEAD
RemoteSha 1720d3371d7921473a93f2ad2db86ce257d16fdf

Contents

autoplot.density_ratios_result	2
autoplot.edp_result	3
autoplot.eta_bias_result	4
autoplot.extrapolation_result	5
autoplot.hat_values_result	6
autoplot.hdr_result	7
autoplot.port_result	8
autoplot.positivity_check	9
check_density_ratios	10
check_edp	12
check_eta_bias	15
check_extrapolation	20
check_hat_values	23
check_hdr	25
check_hdr_seq	28
check_port	30
check_port_seq	32
check_positivity	36
hdr_density_normal	38
new_hdr_density	39
pos_violations	41
pos_violations_long	42
positivity_check	43
positivity_diagnostic	44
pull_plot_data	46
sniff_violations	48
Index	50

autoplot.density_ratios_result
<i>Plot a density-ratio diagnostic</i>

Description

Draws one of two views of a `check_density_ratios()` result. `type = "distribution"` shows the distribution of the raw ratios: a histogram for a point treatment and a per-time boxplot for a time-varying treatment. `type = "cumulative"` shows the cumulative-product quantiles across time points, the sequential positivity signature, and is available for matrix inputs. `pull_plot_data()` returns the tibble either of these views draws.

Arguments

object	A density_ratios_result from <code>check_density_ratios()</code> .
type	One of "distribution" or "cumulative".
...	Not used.

Value

A `ggplot2::ggplot` object.

Examples

```
m <- matrix(c(1, 1, 2, 1, 3, 1, 4, 1), nrow = 2)
result <- check_density_ratios(m)

autoplot(result, type = "distribution")
autoplot(result, type = "cumulative")
```

`autoplot.edp_result` *Plot an effective-data-points diagnostic*

Description

Draws one view of a `check_edp()` result. `type = "boxplot"`, the default, boxes EDP by intervention. Its whiskers reach the fifth and the ninety-fifth percentile, following Ring and Schomaker (2026) rather than Tukey's fences, and the values outside them are drawn as points. For the estimator variant it draws the outcome-model and the treatment-model measure side by side in a facet per intervention. `type = "histogram"` shows the distribution of EDP faceted by intervention, `type = "ecdf"` shows its empirical cumulative distribution colored by intervention, and `type = "density"` draws one density ridgeline per intervention, which needs the **ggridges** package. Each of those three draws one measure: `edp` for the data variant and `edp_outcome` for the estimator variant, the measure carrying the exposure dimension and so the one an intervention moves. `type = "scatter"` plots `edp_outcome` against `edp_treatment` colored by `ideal_weight`; it needs the estimator variant and aborts for the data variant. `pull_plot_data()` returns the tibble any of these views draws.

Arguments

object	An <code>edp_result</code> from <code>check_edp()</code> .
type	One of "boxplot", "histogram", "ecdf", "density", or "scatter".
...	Not used.

Value

A `ggplot2::ggplot` object.

Examples

```
set.seed(1)
n <- 100
x1 <- rnorm(n)
dose <- rnorm(n, mean = x1)
df <- data.frame(dose = dose, x1 = x1)
result <- check_edp(df, dose, x1, values = c(0, 1), exposure_type = "continuous")

autoplot(result)
autoplot(result, type = "histogram")
autoplot(result, type = "ecdf")
```

```
autoplot.eta_bias_result
```

Plot an ETA.Bias diagnostic

Description

Draws one of two views of a `check_eta_bias()` result. `type = "bootstrap"` shows the distribution of bootstrap estimates as a histogram, one facet per estimand term and truncation level, each facet marking the truth its term is aimed at. `type = "sweep"` shows ETA.Bias with a two-Monte-Carlo-standard-error band across the truncation sweep, the bias-variance tradeoff of weight truncation, with one line per estimand term. `pull_plot_data()` returns the tibble either of these views draws.

Arguments

<code>object</code>	An <code>eta_bias_result</code> from <code>check_eta_bias()</code> .
<code>type</code>	One of "bootstrap" or "sweep".
<code>...</code>	Not used.

Details

An estimand of one term, read at one truncation level, draws a single facet. A discrete exposure truncates a fitted probability, so its sweep is drawn against the truncation lower bound. A continuous exposure caps a stabilized weight instead and reports no probability bound at any level, so its sweep is drawn against the quantile levels those caps were read at.

Value

A `ggplot2::ggplot` object.

Examples

```

set.seed(1)
n <- 300
x1 <- rnorm(n)
x2 <- rnorm(n)
a <- rbinom(n, 1, plogis(2 * (x1 + x2)))
y <- a + x1 + x2 + rnorm(n)
df <- data.frame(a = a, y = y, x1 = x1, x2 = x2)
# n_boot is small here to keep the example fast.
result <- check_eta_bias(df, a, y, c(x1, x2), n_boot = 25)
autoplot(result, type = "bootstrap")

swept <- check_eta_bias(
  df,
  a,
  y,
  c(x1, x2),
  truncation_grid = c(0, 0.05, 0.1),
  n_boot = 25
)
autoplot(swept, type = "sweep")

# A continuous exposure caps its weights at a quantile, so the sweep reads
# the quantile levels rather than a probability bound.
df$a <- x1 + x2 + rnorm(n)
df$y <- df$a + x1 + x2 + rnorm(n)
continuous <- check_eta_bias(
  df,
  a,
  y,
  c(x1, x2),
  truncation_grid = c(0, 0.05, 0.1),
  n_boot = 25
)
autoplot(continuous, type = "sweep")

```

autoplot.extrapolation_result

Plot an extrapolation diagnostic

Description

Draws one of two views of a `check_extrapolation()` result. `type = "distribution"` shows the distribution of `frac_nearby` as histograms faceted by exposure group, the share of the opposite group that lies nearby each observation. `type = "hull"` shows convex-hull membership as stacked bars per exposure group, and is available only when the hull test ran. `pull_plot_data()` returns the tibble either of these views draws.

Arguments

object	An extrapolation_result from <code>check_extrapolation()</code> .
type	One of "distribution" or "hull".
...	Not used.

Value

A `ggplot2::ggplot` object.

Examples

```
set.seed(1)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
exposure <- rbinom(n, 1, plogis(0.5 * x1 - 0.5 * x2))
df <- data.frame(exposure = exposure, x1 = x1, x2 = x2)
result <- check_extrapolation(df, exposure, c(x1, x2), hull = FALSE)

autoplot(result, type = "distribution")
```

autoplot.hat_values_result

Plot a hat-value diagnostic

Description

Draws one of two views of a `check_hat_values()` result. `type = "null"` shows the null distribution of $\hat{\phi}$ as a histogram with the observed $\hat{\phi}$ and the `conf_level` null quantile marked. `type = "profile"` shows the proportion of high-leverage candidates at each exposure percentile, the leverage profile across the exposure range. `pull_plot_data()` returns the tibble either of these views draws.

Arguments

object	A hat_values_result from <code>check_hat_values()</code> .
type	One of "null" or "profile".
...	Not used.

Value

A `ggplot2::ggplot` object.

Examples

```

set.seed(1)
n <- 200
x1 <- rnorm(n)
dose <- rnorm(n, mean = x1)
df <- data.frame(dose = dose, x1 = x1)
# null_reps is small here to keep the example fast.
result <- check_hat_values(df, dose, x1, null_reps = 50)

autoplot(result, type = "null")
autoplot(result, type = "profile")

```

autoplot.hdr_result *Plot an HDR non-overlap diagnostic*

Description

Draws the non-overlap ratio $\hat{\tau}(a)$ against the target exposure value a , with a reference line at zero. For a sequential result the curves are colored by time point. [pull_plot_data\(\)](#) returns the tibble this view draws.

Arguments

object	An <code>hdr_result</code> from check_hdr() or check_hdr_seq() .
...	Not used.

Value

A [ggplot2::ggplot](#) object.

Examples

```

set.seed(1)
n <- 300
l <- rnorm(n)
dose <- rnorm(n, mean = l)
df <- data.frame(dose = dose, l = l)
result <- check_hdr(df, dose, l)

autoplot(result)

```

autoplot.port_result *Plot a PoRT subgroup diagnostic*

Description

Draws the reported subgroups as a horizontal bar chart of exposure prevalence, with dashed reference lines at β and $1 - \beta$. Bar width is proportional to subgroup size, and low-support subgroups are highlighted. A sequential result is faceted by time point, and each facet carries its own resolved thresholds. A censoring run is faceted by both time point and row type, so the exposure and censoring subgroups are separated, the censoring facets plot censoring prevalence, and each facet's reference lines are drawn from the thresholds that family was judged against. [pull_plot_data\(\)](#) returns the tibble this view draws.

Arguments

object	A port_result from check_port() or check_port_seq() .
low_support_only	If TRUE, draw only the low-support subgroups. Defaults to FALSE, which draws every reported subgroup.
...	Not used.

Details

A full result can report hundreds of subgroups, which crowds the y axis. Setting `low_support_only = TRUE` draws only the low-support subgroups and drops the fill legend, while the facets and reference lines are kept, so a facet without bars means nothing read as low support at that time point.

Value

A [ggplot2::ggplot](#) object.

Examples

```
set.seed(1)
n <- 400
x1 <- rnorm(n)
exposure <- rbinom(n, 1, ifelse(x1 > 1, 0, 0.5))
df <- data.frame(exposure = exposure, x1 = x1)
result <- check_port(df, exposure, x1)

autoplot(result)
autoplot(result, low_support_only = TRUE)
```

autoplot.positivity_check

Plot the diagnostics a positivity check holds

Description

Draws each child's default view. With no diagnostic, the views are composed into one panel; naming a diagnostic draws that child alone.

Arguments

object	A positivity_check from check_positivity() .
diagnostic	Optionally, the name of one diagnostic to draw. Defaults to NULL, which panels every child.
...	Passed to a child's own autoplot() method. The panel passes them to every child, so an argument only one diagnostic accepts belongs with a named diagnostic rather than with the panel.

Details

The panel needs the **patchwork** package, which composes the children into a single object that is still a [ggplot2::ggplot](#) and so can be built on further. Without it, naming a diagnostic still works, because that path draws one child's plot and composes nothing.

A diagnostic is named the way [names\(\)](#) lists it and [\\$](#) extracts it, which is also the name the printed report heads its sections with, so a reader can go from a section straight to its plot.

Value

A [ggplot2::ggplot](#) object.

Examples

```
set.seed(1)
n <- 300
x1 <- rnorm(n)
df <- data.frame(exposure = rbinom(n, 1, plogis(3 * x1)), x1 = x1)

check <- check_positivity(df, exposure, x1, diagnostics = c("port", "edp"))

autoplot(check)
autoplot(check, "port")
```

check_density_ratios *Summarize density ratios for positivity diagnosis*

Description

check_density_ratios() summarizes user-supplied density ratios, the Radon-Nikodym derivatives that arise as weights in modified treatment policy and longitudinal causal analyses. It is a bring-your-own-numbers diagnostic: a numeric vector is treated as a point treatment, an n-by-T matrix as a time-varying treatment whose cumulative products across time points are summarized alongside the per-time ratios, and an **lmtp** fit is read for its density-ratio component.

Usage

```
check_density_ratios(ratios, ...)

## S3 method for class 'numeric'
check_density_ratios(
  ratios,
  probs = c(0.5, 0.9, 0.95, 0.99, 1),
  thresholds = c(10, 50),
  ...
)

## S3 method for class 'matrix'
check_density_ratios(
  ratios,
  probs = c(0.5, 0.9, 0.95, 0.99, 1),
  thresholds = c(10, 50),
  ...
)
```

Arguments

ratios	The density ratios. A numeric vector for a point treatment (a propensity::psw object is accepted and read through vctrs::vec_data()), an n-by-T numeric matrix whose columns are time points for a time-varying treatment, or a fitted lmtp object whose density-ratio component is read out. Ratios must be non-negative, finite, and free of missing values.
...	Passed to methods.
probs	A numeric vector of quantile probabilities in $[0, 1]$. Defaults to <code>c(0.5, 0.9, 0.95, 0.99, 1)</code> ; the 1 quantile is reported as the maximum.
thresholds	A numeric vector of positive exceedance thresholds. Defaults to <code>c(10, 50)</code> .

Details

A density ratio is $r = g^*(a | h)/g(a | h)$, the intervention treatment law relative to the observed treatment law. It is the weight that reweights the observed data to the interventional world, and it is large exactly where the observed conditional treatment density is small, that is, where positivity is threatened. Petersen and colleagues (2012) recommend inspecting the distribution of these weights, because near violations of positivity surface as a heavy upper tail, while values of the observed density close to zero for some treatment level signal a violation directly.

Because a density ratio is a Radon-Nikodym derivative, its mean is one by construction whenever the intervention law is absolutely continuous with respect to the observed law. The mean is therefore a specification check, not a violation detector: well-behaved weights alone do not guarantee positivity. The signal instead lives in the upper tail (the quantiles, the maximum, and the proportion of ratios above the thresholds) and in the Kish effective sample size, which collapses as a few large weights come to dominate. A mean visibly below one is itself informative: it marks a structural violation in which intervention probability mass leaks outside the observed support, so that some ratios are exactly zero.

For each time point the diagnostic reports the mean, the quantiles at probs (named `quantile_50`, `quantile_90`, and so on, with the `probs = 1` quantile reported as `max`), the proportion of ratios exceeding each of thresholds (named `prop_gt_10`, `prop_gt_50`), the proportion of exact-zero ratios (`prop_zero`), the Kish effective sample size `ess`, and its fraction of the sample size `ess_fraction`. The Kish effective sample size is $(\sum_i r_i)^2 / \sum_i r_i^2$; it equals the sample size when the ratios are equal and is invariant to their overall scale.

For a matrix input the same summaries are also applied to the cumulative product of the ratios across time points, under the `cumulative_` prefix. This is the sequential positivity signature: mild per-step non-overlap compounds multiplicatively, so the cumulative effective sample size can collapse even while each per-time fraction stays healthy. For a single-column matrix the cumulative summaries coincide with the per-time summaries. The default thresholds of 10 and 50 are conventional weight-magnitude heuristics rather than values sanctioned by either source paper.

Value

A `density_ratios_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble is long, with columns `time` (the time point, 1 for a point treatment), `statistic` (the summary name), and `value`. It also carries the raw per-time ratios as the list property `@ratios`, with one element per time point.

`generics::glance()` returns a one-row tibble with `n` (the sample size), `n_times`, and the headline statistics of the cumulative ratios at the final time point: `mean`, `max`, `prop_zero`, and `ess_fraction`. For a point treatment those are the statistics of the supplied ratios.

References

- Petersen ML, Porter KE, Gruber S, Wang Y, van der Laan MJ (2012). Diagnosing and responding to violations in the positivity assumption. *Statistical Methods in Medical Research*, 21(1):31-54. doi:10.1177/0962280210386207
- Ring C, Schomaker M (2026). A Diagnostic to Find and Help Combat Stochastic Positivity Issues, with a Focus on Continuous Treatments.

Examples

```
# Point treatment: a numeric vector of density ratios.
ratios <- c(0.5, 1, 1, 2, 8)
check_density_ratios(ratios)

# Time-varying treatment: one column per time point. The cumulative products
# across columns are summarized alongside the per-time ratios.
m <- matrix(c(1, 1, 2, 1, 3, 1), nrow = 2)
check_density_ratios(m)
```

check_edp

Diagnose positivity with effective data points

Description

check_edp() computes the effective-data-points (EDP) positivity diagnostic of Ring and Schomaker (2026) for binary, categorical, and continuous exposures. For every observation moved to a candidate intervention value, it sums a product kernel over the sample to measure how much observed support surrounds that intervened-on point.

Usage

```
check_edp(
  .data,
  .exposure,
  .covariates,
  .outcome_covariates = NULL,
  .treatment_covariates = NULL,
  values = NULL,
  variant = c("data", "estimator"),
  kernel = c("gaussian"),
  bw_exposure = NULL,
  bw_covariates = NULL,
  categorical_similarity = 0,
  exposure_type = c("auto", "binary", "categorical", "continuous")
)
```

Arguments

.data	A data frame.
.exposure	The exposure column, selected with data-masking.
.covariates	The covariate columns, selected with tidyselect.
.outcome_covariates	The outcome-model covariate columns for the estimator variant, selected with tidyselect. Defaults to .covariates.

<code>.treatment_covariates</code>	The treatment-model covariate columns for the estimator variant, selected with <code>tidyselect</code> . Defaults to <code>.covariates</code> .
<code>values</code>	The interventions, one per element. NULL (the default) uses every level of a factor exposure and every observed value of a binary or character exposure, and the deciles of the observed exposure for continuous exposures. A vector holds one intervention value a^* per element, the value every observation is moved to. A list holds the same single values and, for a continuous exposure, functions of the observed exposure: <code>list(3, 5, lower = function(.x) .x - 5)</code>. A function is called once on the whole exposure vector and returns one intervened exposure $d(a_i)$ per observation, or a single value used for every observation. Functions need a continuous exposure; one supplied for a binary or categorical exposure is an error. Every intervention carries a label, which names it in the results and along a plot's axis. In a vector, every element is labeled by <code>as.character()</code> of the value itself and any names the vector carries are ignored. In a list, a named element is labeled by its name, an unnamed single value by <code>as.character()</code> of the value itself, and an unnamed function is an error, since it has nothing else to be called by. Two elements of a list cannot share a label.
<code>variant</code>	One of "data" (report a single edp per point, the default) or "estimator" (report <code>edp_outcome</code> , <code>edp_treatment</code> , and <code>ideal_weight</code>).
<code>kernel</code>	The continuous-dimension kernel. Only "gaussian" is currently available; categorical dimensions always use the match kernel.
<code>bw_exposure</code>	The continuous-exposure half-distance. NULL (the default) uses $0.5 * \text{sd}(\text{exposure})$. A half-distance of 0 counts exact matches.
<code>bw_covariates</code>	The numeric-covariate half-distance, applied to every numeric covariate. NULL (the default) uses $1 * \text{sd}$ per covariate.
<code>categorical_similarity</code>	The kernel value in $[0, 1]$ for non-matching categories, shared by categorical exposures and factor or character covariates. Defaults to 0. A positive value masks categorical violations.
<code>exposure_type</code>	One of "auto" (detect from the data, the default), "binary", "categorical", or "continuous". A supplied type is authoritative and detection is not consulted, so it is rejected only when the exposure column cannot carry it: "continuous" needs a numeric column and "binary" needs exactly two distinct values, while "categorical" asks nothing of the column.

Details

For each observation i the diagnostic forms the intervened-on point $o_i^* = (l_i, a^*)$, holding the covariates at their observed values and setting the exposure to an intervention value a^* . The effective data points at that point are

$$\text{EDP}_i(a^*) = \sum_{j=1}^n k(o_j, o_i^*),$$

a sum over the whole sample of a product kernel that multiplies one factor per dimension. Each factor lies in $[0, 1]$, so $0 \leq \text{EDP} \leq n$. A continuous dimension with half-distance h contributes

$0.5^{(\delta/h)^2}$, which equals 1 at $\delta = 0$ and exactly 0.5 at $|\delta| = h$; this is a reparameterized Gaussian kernel. A categorical dimension contributes 1 on a match and categorical_similarity otherwise. Numeric covariates and continuous exposures are continuous dimensions; factor and character covariates, together with binary and categorical exposures, are categorical dimensions.

An intervention need not be a single value every observation is moved to. A function supplied through values is a rule d applied to the observed exposure, so observation i is moved to $d(a_i)$ and the intervened-on point is $o_i^* = (l_i, d(a_i))$. Lowering every observed dose by five units is `function(.x) .x - 5`. EDP is then read per natural observation under the shift rather than at one common target; the algorithm is unchanged, and only the intervened-on points come from the rule. Because those points differ across observations, the exposure factor of the kernel compares every observed exposure against a target that moves with the observation rather than against a single a^* . A run therefore holds one $n \times n$ matrix of exposure-kernel values per function intervention, materialized before any summing and held for the duration of the run.

The half-distances follow the paper's rule of thumb: $1 * \text{sd}$ for each numeric covariate and $0.5 * \text{sd}$ for a continuous exposure. These are marginal scales, so a mid-support gap in a strongly multimodal exposure can be masked by an inflated standard deviation; supply `bw_exposure` or `bw_covariates` directly when a robust scale is preferable. As the half-distances grow without bound every kernel factor approaches 1 and EDP approaches n , which renders the diagnostic uninformative; at a half-distance of 0 a continuous dimension counts only exact matches.

EDP is a product kernel, so adding dimensions can only lower it. It is meaningful relative across observations and strata for a fixed covariate set, never against a universal threshold; compare a stratum against another stratum or a candidate value against another candidate value rather than against a fixed cutoff.

The estimator variant reframes EDP around the two nuisance models of a causal estimator. `edp_outcome` conditions on the exposure and the outcome-model covariates (A, L) ; `edp_treatment` conditions on the treatment-model covariates (L) alone. Because the outcome measure carries the extra exposure dimension, `edp_outcome <= edp_treatment` whenever the two covariate sets are equal. The reported `ideal_weight` is `edp_treatment / edp_outcome`, proportional to the inverse of the estimated treatment density at the intervention, so it grows where treatment support is thin. Where a target has no outcome-model support, `edp_outcome` is zero and `ideal_weight` is infinite, the honest value of one over an estimated density of zero.

Value

An `edp_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble has one row per observation and intervention. The data variant carries columns `.id` (the observation row index), `intervention` (the intervention label), `value` (the intervened exposure, the constant a^* for a single value and $d(a_i)$ for a function), and `edp`. The estimator variant replaces `edp` with `edp_outcome`, `edp_treatment`, and `ideal_weight`. It also carries the `@variant` and `@bandwidths` properties.

`generics::glance()` returns a one-row tibble with `n` (the sample size), `variant`, `n_values` (the number of interventions), and the observed range of every measure the variant reports. That is `edp_min` and `edp_max` for the data variant, and `edp_outcome_min`, `edp_outcome_max`, `edp_treatment_min`, `edp_treatment_max`, `ideal_weight_min`, and `ideal_weight_max` for the estimator variant.

References

Ring C, Schomaker M (2026). A Diagnostic to Find and Help Combat Stochastic Positivity Issues, with a Focus on Continuous Treatments.

Examples

```

set.seed(1)
n <- 100
x1 <- rnorm(n)
dose <- rnorm(n, mean = x1)
df <- data.frame(dose = dose, x1 = x1)

result <- check_edp(df, dose, x1, values = c(0, 1), exposure_type = "continuous")
result

# An intervention can also be a function of the observed exposure, read as a
# shift away from the dose each observation received
shifted <- check_edp(
  df,
  dose,
  x1,
  values = list(0, 1, lower = function(.x) .x - 1),
  exposure_type = "continuous"
)
tidy(shifted)

```

check_eta_bias

*Diagnose positivity bias with the parametric bootstrap (ETA.Bias)***Description**

check_eta_bias() implements the ETA.Bias parametric-bootstrap diagnostic of Petersen et al. (2012) for binary, categorical, and continuous point exposures. It estimates the bias that a chosen causal estimator would incur under the data's own fitted treatment and outcome mechanisms, isolating the component of bias that stems from positivity violations, weight truncation, and finite-sample sparsity.

Usage

```

check_eta_bias(
  .data,
  .exposure,
  .outcome,
  .covariates,
  estimator = c("ipw", "gcomp", "aipw"),
  exposure_formula = NULL,
  outcome_formula = NULL,
  outcome_type = c("auto", "continuous", "binary"),
  truncation = NULL,
  truncation_grid = NULL,
  n_boot = 500,
  error_dist = c("normal", "empirical"),

```

```

exposure_type = c("auto", "binary", "categorical", "continuous"),
reference_level = NULL,
msm_formula = NULL
)

```

Arguments

<code>.data</code>	A data frame.
<code>.exposure</code>	The exposure column, selected with data-masking. A binary exposure has exactly two distinct values; a categorical exposure has as many levels as it has distinct values, and every non-reference level is contrasted against the reference; a continuous exposure has no levels, and is summarized by the working model <code>msm_formula</code> sets.
<code>.outcome</code>	The outcome column, selected with data-masking.
<code>.covariates</code>	The covariate columns, selected with <code>tidyselect</code> . Numeric, logical, factor, and character covariates are all supported. The fitted treatment and outcome models expand factor and character covariates into indicator terms, and the bootstrap resamples covariate rows, so factor levels are preserved throughout.
<code>estimator</code>	The causal estimator to diagnose, either one of the built-in estimators or one of your own. The built-in estimators are named by a string: "ipw" (inverse probability weighting, matching <code>propensity::ipw()</code>), "gcomp" (G-computation), or "aipw" (augmented, doubly robust). The augmented estimator's correction is a sum over the exposure levels, so it is an error for a continuous exposure. An estimator of your own is a function, or a length-one list whose name labels that function. A bare function is labeled "custom". The label is what <code>@estimator</code>, <code>@params</code>, and <code>generics::glance()</code> report; the function itself is not kept on the result. The function is called once per bootstrap draw and receives one argument, the data frame of that draw: the resampled covariate rows, the exposure drawn from the fitted treatment mechanism, and the outcome drawn from the fitted outcome regression, under the column names they were selected by. A discrete exposure arrives as a factor over the levels the observed exposure holds, and a continuous one as the drawn numeric vector. The function must return one number per estimand term: named by the terms, in which case the values are read by name, or unnamed, in which case they are read in term order. Every draw's return is read on its own, so the two forms may be mixed from one draw to the next, and names that are anything other than the estimand terms are an error wherever they appear. The function has to aim at the estimand the run's terms define, which is the set of contrasts against <code>reference_level</code> for a discrete exposure and the coefficients of <code>msm_formula</code> for a continuous one. The truth is the same G-computation target whichever estimator is diagnosed, so an estimator aimed at anything else reports an estimand mismatch as part of its bias. <code>truncation</code> and <code>truncation_grid</code> bound a fitted nuisance model the function never receives, so supplying either alongside one is an error. A draw the function

raises an error on is dropped exactly as a non-finite estimate is, and a run whose every draw was dropped is an error.

exposure_formula	An optional model formula for the treatment mechanism. NULL (the default) fits a main-effects model of the exposure on the covariates, logistic at two levels, a multinomial logit past two, and a normal linear model for a continuous exposure.
outcome_formula	An optional model formula for the outcome regression. NULL (the default) fits a main-effects model of the outcome on the exposure and covariates. Terms that recompute a data-dependent basis from the exposure column (for example <code>poly()</code> of the exposure) are not supported, because the counterfactual predictions rebuild the design with the exposure set to each of its levels in turn.
outcome_type	One of "auto" (detect from the outcome, the default), "continuous", or "binary".
truncation	An optional length-two numeric vector <code>c(lower, upper)</code> bounding the fitted propensity for a single run. NULL (the default) applies no truncation. A pair of bounds describes the one-dimensional simplex of a two-level exposure, so it applies to binary exposures only and is an error past two levels or on a continuous exposure.
truncation_grid	An optional numeric vector, each entry in $[0, 1/k)$ for an exposure with k levels, which is $[0, 0.5)$ for a binary exposure, and in $[0, 1)$ for a continuous one. All levels share one set of bootstrap draws, and the grid overrides truncation. What an entry means depends on the exposure type: at two levels it is a lower bound and becomes the swept truncation level <code>c(lower, 1 - lower)</code> ; past two levels it raises every fitted probability below it and renormalizes each row, leaving no upper bound to apply or report; and for a continuous exposure it is a quantile level g that caps the stabilized weight at its $1 - g$ quantile, read once from the observed fit and held fixed across the bootstrap draws.
n_boot	The number of bootstrap datasets. Defaults to 500. Must be at least 2, since the Monte Carlo standard error needs two draws.
error_dist	The bootstrap error model for continuous outcomes, one of "normal" (add Gaussian noise matched to the residual standard deviation, the default) or "empirical" (resample residuals). Ignored for binary outcomes.
exposure_type	One of "auto" (detect from the data, the default), "binary", "categorical", or "continuous". A supplied type is authoritative and detection is not consulted, so it is rejected only when the exposure column cannot carry it: "binary" needs exactly two distinct values, "continuous" needs a numeric column, and "categorical" needs nothing beyond the two levels every discrete run needs. Within the discrete types the run is decided by the number of levels, so declaring "categorical" on a two-level column returns the binary run unchanged.
reference_level	The exposure level every contrast is taken against. NULL (the default) takes the first level, which is the lower of the two values for a binary exposure and the first factor level otherwise. A continuous exposure has no levels to contrast, so supplying it there is an error.

`msm_formula` An optional one-sided formula for the marginal structural model a continuous exposure is summarized by, for example $\sim a + I(a^2)$. NULL (the default) fits a model linear in the exposure, whose one coefficient is named after the exposure column. A discrete estimand is a set of contrasts and imposes no working model, so supplying it there is an error.

Details

The diagnostic (the paper’s Section 4.1) treats the fitted data-generating mechanism as a known truth and asks how a candidate estimator behaves under it. The algorithm has three steps: fit the treatment and outcome mechanisms, declare the fitted mechanism the truth, then resample from it and refit. The exposure type changes what each step does, and nothing about the order they come in.

First, it fits the treatment mechanism g_n and the outcome regression $\bar{Q}_n(A, W)$, a main-effects linear or logistic model of the outcome on the exposure and the covariates. Factor and character covariates enter both models as indicator terms, exactly as `stats::glm()` expands them. The treatment mechanism is a main-effects model of the exposure on the covariates in the form its type calls for: the logistic model $g_n(1 | W)$ at two levels; the multinomial logit $g_n(a | W)$ past two, which requires the **nnet** package; and for a continuous exposure the normal linear model $A | W \sim \mathcal{N}(m_n(W), \sigma_n^2)$, whose spread is the pooled maximum-likelihood residual standard deviation $\sigma_n = \sqrt{n^{-1} \sum_i (A_i - m_n(W_i))^2}$.

The truth is the target the fitted mechanism implies, which is available exactly because that fit is the truth of the bootstrap data-generating mechanism. It depends on the outcome model alone, so it is identical across estimators and constant across truncation levels. A discrete exposure is summarized by the saturated factor model, whose coefficients are the G-computation contrasts against the reference level a_0 , $\psi_a = \frac{1}{n} \sum_i [\bar{Q}_n(a, W_i) - \bar{Q}_n(a_0, W_i)]$, one per non-reference level a . A saturated model restricts nothing, so that summary imposes nothing; an exposure coded 0/1 is the two-level case and its one contrast is ψ_1 . A continuous exposure has no levels to saturate. Its counterfactual mean curve $\frac{1}{n} \sum_i \bar{Q}_n(a, W_i)$ is evaluated over the unique deciles of the observed exposure and projected by least squares onto the working model `msm_formula` sets, and the truth is the coefficients of that projection of the observed fit. Equal weights on a decile grid weight the projection by the marginal density of the exposure.

Second, it draws `n_boot` bootstrap datasets. Each resamples the covariate rows with replacement, draws a bootstrap exposure A^* from the untruncated fitted treatment mechanism at the resampled covariates, and draws a bootstrap outcome from $\bar{Q}_n(A^*, W^*)$: for continuous outcomes by adding normal or empirical-residual error, and for binary outcomes by a Bernoulli draw. The exposure draw is $A^* \sim \text{Bernoulli}(g_n(1 | W^*))$ at two levels, a categorical draw over the fitted probabilities $g_n(a | W^*)$ past two, and $A^* \sim \mathcal{N}(m_n(W^*), \sigma_n^2)$ for a continuous exposure. Truncation is a property of the estimator, not of the data-generating mechanism, so the bootstrap exposure is always drawn from the untruncated mechanism.

Third, it refits both nuisance models on each bootstrap dataset and applies the candidate estimator, truncating what that estimator divides by. `ETA.Bias` is the mean of the bootstrap estimates minus truth, and its Monte Carlo standard error is the standard deviation of the bootstrap estimates divided by the square root of the number of retained draws. A bootstrap draw whose estimate is non-finite carries no information, and is dropped with a warning before the summaries are formed.

What a `truncation_grid` entry g truncates also follows the exposure type. At two levels it bounds the fitted propensity into $[g, 1 - g]$: the simplex is one-dimensional there, so a lower bound on one

level is an upper bound on the other and the rule is two-sided. Past two levels it raises every fitted probability below g up to g and renormalizes each row to sum to one, which is the matrix method of `propensity::ps_trunc()` under method = "ps". A lower pin on every level already bounds every weight denominator, so no upper bound is applied and none is reported. A continuous exposure divides by a density rather than by a probability, and g caps the stabilized weight $f_n(A)/f_n(A | W)$ at its $1 - g$ quantile, read once from the observed fit and held fixed across the bootstrap draws.

The three estimators behave differently under positivity violations. G-computation ignores the treatment mechanism entirely, so its ETA.Bias is Monte Carlo noise around zero and is flat across a truncation grid. Inverse probability weighting divides by the fitted mechanism, so extreme probabilities and heavy density ratios inflate its ETA.Bias, and tightening the truncation bound trades bias for variance. The augmented (doubly robust) estimator stays near zero in bias while carrying more variance than G-computation, and it is defined for a discrete estimand alone.

When `truncation_grid` is supplied, every grid point reuses one shared set of bootstrap draws and nuisance refits, so the sweep isolates the effect of the truncation bound alone: the inverse-probability-weighting bias rises and its bootstrap spread shrinks as the bound tightens.

A continuous estimand is defined by the working model rather than approximated by it, so two analyses of the same data under different `msm_formula` values are aimed at different targets and report different biases. That is a property of the estimand and not an inconsistency: the bias reported is the bias of estimating the working model the analysis actually uses.

ETA.Bias captures only the positivity, truncation, and sparsity component of bias and excludes model-misspecification bias by construction. It is a red flag for a fitted mechanism, not a bias correction.

Value

An `eta_bias_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble has one row per estimand term and truncation level, with the columns `term`, `truncation_lower`, `truncation_upper`, `bias` (ETA.Bias), `mc_se` (the Monte Carlo standard error), and `boot_mean` (the mean bootstrap estimate), in that order. `term` names the estimand contrast as `<level> - <reference level>`, so an exposure coded 0/1 has the single term `"1 - 0"`; for a continuous exposure it names a working-model coefficient instead. A continuous run has no lower cap to place on a density ratio, so `truncation_lower` is `NA_real_` and `truncation_upper` holds the weight cap, which is `Inf` where no cap was applied. The object also carries the properties `@estimator`, `@truth` (a named numeric vector holding one truth per term), and `@boot_estimates` (a list of bootstrap-estimate vectors, one per row of `@results`).

`generics::glance()` returns a one-row tibble with `n` (the sample size), `estimator`, and `n_boot`. A run that dropped a bootstrap draw adds `n_boot_used`, the number of draws retained, counted as the smallest number any reported row was formed from; a run that dropped none omits the column. A single-term estimand adds `truth`; an estimand of more than one term has no single truth, so it reports `n_terms` in its place. A one-row result then adds `bias` and `mc_se`, while a result of more rows replaces those two with `n_levels`, `bias_min`, and `bias_max`. The row count is what decides this, not the argument used, so a length-one `truncation_grid` takes the one-row form.

References

Petersen ML, Porter KE, Gruber S, Wang Y, van der Laan MJ (2012). Diagnosing and responding to violations in the positivity assumption. *Statistical Methods in Medical Research*, 21(1):31–54. doi:10.1177/0962280210386207

Examples

```
set.seed(1)
n <- 300
x1 <- rnorm(n)
x2 <- rnorm(n)
a <- rbinom(n, 1, plogis(x1 + x2))
y <- a + x1 + x2 + rnorm(n)
df <- data.frame(a = a, y = y, x1 = x1, x2 = x2)

# n_boot is small here to keep the example fast.
check_eta_bias(df, a, y, c(x1, x2), estimator = "ipw", n_boot = 25)

# An estimator of your own reads one bootstrap dataset per draw. The exposure
# arrives as a factor of its own levels, so the fitted coefficient is named
# for the level it contrasts.
by_hand <- function(.data) {
  coef(lm(y ~ a + x1 + x2, data = .data))["a1"]
}
check_eta_bias(
  df,
  a,
  y,
  c(x1, x2),
  estimator = list(gformula = by_hand),
  n_boot = 25
)
```

check_extrapolation	<i>Diagnose extrapolation for binary exposures with Gower distances and the convex hull</i>
---------------------	---

Description

check_extrapolation() implements the extrapolation diagnostics of King and Zeng (2006) for binary point exposures. It measures, for every observation, how well the opposite exposure group covers its position in covariate space, using Gower distances that handle mixed numeric and categorical covariates, and it optionally tests convex-hull membership against the opposite group.

Usage

```
check_extrapolation(
  .data,
  .exposure,
  .covariates,
  nearby = 1,
  hull = NULL,
  exposure_type = c("auto", "binary")
)
```

Arguments

<code>.data</code>	A data frame.
<code>.exposure</code>	The binary exposure column, selected with data-masking. <code>check_extrapolation()</code> aborts unless the exposure has exactly two distinct values, whether that type is detected or declared through <code>exposure_type</code> .
<code>.covariates</code>	The covariate columns, selected with <code>tidyselect</code> . Numeric and categorical covariates are both supported by the Gower distance.
<code>nearby</code>	The reference radius as a multiple of the geometric variability. An observation counts an opposite-group unit as nearby when their Gower distance is at most <code>nearby</code> times the geometric variability. A single positive number, defaulting to 1.
<code>hull</code>	Whether to run the convex-hull membership test. <code>NULL</code> (the default) runs it automatically when there is at least one and at most ten numeric covariates and lpSolve is installed, and skips it otherwise. <code>TRUE</code> forces it to run, aborting when lpSolve is not installed and warning when there are no numeric covariates or the dimension is high. <code>FALSE</code> skips it.
<code>exposure_type</code>	One of "auto" (detect from the data, the default) or "binary". A supplied type is authoritative and detection is not consulted, so "binary" is rejected only when the exposure column does not have exactly two distinct values. Declaring it unlocks no call that "auto" would refuse, since detection reads any two-valued column as binary; it is accepted so that every diagnostic takes the same argument.

Details

Estimating a causal contrast requires comparing treated and untreated units with similar covariates. Where one exposure group has no nearby members of the other, the contrast rests on extrapolation rather than data, the concern framed by Petersen and colleagues (2012) and by King and Zeng (2006).

The Gower distance between two rows is the mean over covariates of a per-covariate distance: for a numeric covariate, the absolute difference divided by the pooled sample range; for a factor or character covariate, one when the categories differ and zero when they agree. Distances therefore lie in $[0, 1]$ and mixed covariate types combine on a common scale. Because the denominator is the sample range, a single extreme covariate value rescales every distance on that axis.

The geometric variability is one half the mean of the full Gower distance matrix, the Cuadras-Arenas quantity used by the WhatIf software. It sets the reference radius: an observation is counted as nearby-supported when an opposite-group unit lies within nearby geometric variabilities. Three per-unit statistics are reported: `gower_min` (distance to the nearest opposite-group unit), `gower_mean` (mean distance to the opposite group), and `frac_nearby` (the fraction of the opposite group within nearby geometric variabilities). A planted support gap raises `gower_min` and lowers `frac_nearby` in both directions. The per-group `prop_supported` reported by `summary()`, the `prop_supported` statistic in `glance()`, and the print method share one fixed definition: the fraction of units whose nearest opposite-group unit lies within one geometric variability (`gower_min <= geometric_variability`), independent of `nearby`. The per-unit `low_support` column is the complement of that condition, so `prop_supported` is the fraction of rows whose `low_support` is `FALSE`.

When every covariate is constant, all rows coincide, the geometric variability is zero, and each observation is at distance zero from the opposite group. Such an observation is fully supported: `gower_min` is zero and `frac_nearby` is one. Constant numeric covariates likewise contribute zero distance on their own axis without affecting the others.

The convex-hull test asks whether an observation lies inside the convex hull of the opposite group, solved as a linear-programming feasibility problem (weights that are non-negative, sum to one, and reproduce the point). The hull test uses the numeric covariates only, since categorical covariates carry no ordering to interpolate over. In-hull membership collapses toward zero as the covariate dimension grows even under perfect overlap, so the test is degenerate in high dimensions (D’Amour and colleagues, 2021). It therefore runs automatically only when there is at least one and at most ten numeric covariates and **lpSolve** is installed. With no numeric covariates, more than ten of them, or **lpSolve** absent, it is skipped and `in_hull` is NA. Setting `hull = TRUE` forces the test to run and warns when the dimension is high; `hull = FALSE` skips it.

Value

An `extrapolation_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble has one row per observation with columns `.id` (the row index), `exposure` (the exposure value), `frac_nearby`, `gower_min`, `gower_mean`, `in_hull` (the hull membership flag, or NA when the hull test did not run), and `low_support` (TRUE when the nearest opposite-group unit lies farther away than one geometric variability). It also carries the scalar properties `@geometric_variability` and `@hull_run`.

`summary()` aggregates the results by exposure group, returning one row per exposure level with the columns `exposure`, `n`, `mean_gower_min`, `prop_supported` (the one geometric variability fraction described in Details), and `prop_in_hull` (the fraction inside the opposite group’s hull, NA when the hull test did not run).

`generics::glance()` returns a one-row tibble with `n` (the sample size), `nearby`, `geometric_variability`, `mean_frac_nearby`, `prop_supported`, `hull_run` (kept logical), and `prop_in_hull`, which is NA when the hull test did not run. The two readings, `prop_supported` and `prop_in_hull`, answer different questions and are both reported.

References

- King G, Zeng L (2006). The Dangers of Extreme Counterfactuals. *Political Analysis*, 14(2):131-159. doi:10.1093/pan/mpj004
- Petersen ML, Porter KE, Gruber S, Wang Y, van der Laan MJ (2012). Diagnosing and responding to violations in the positivity assumption. *Statistical Methods in Medical Research*, 21(1):31-54. doi:10.1177/0962280210386207
- D’Amour A, Ding P, Feller A, Lei L, Sekhon J (2021). Overlap in Observational Studies with High-Dimensional Covariates. *Journal of Econometrics*, 221(2):644-654. doi:10.1016/j.jeconom.2019.10.014

Examples

```
set.seed(1)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
exposure <- rbinom(n, 1, plogis(0.5 * x1 - 0.5 * x2))
```

```
df <- data.frame(exposure = exposure, x1 = x1, x2 = x2)

# hull = FALSE keeps the example free of the optional lpSolve dependency.
result <- check_extrapolation(df, exposure, c(x1, x2), hull = FALSE)
result
```

check_hat_values

*Diagnose positivity for continuous exposures with hat values***Description**

check_hat_values() computes the hat-value (leverage) positivity diagnostic of Moodie and Schulz (2025) for continuous exposures. It measures how far a set of candidate exposure-covariate combinations sits from the observed data by the leverage those combinations would exert on a linear model, then compares the observed leverage profile against a null in which the exposure is drawn independently of the covariates.

Usage

```
check_hat_values(
  .data,
  .exposure,
  .covariates,
  probs = seq(0.05, 0.95, by = 0.05),
  threshold = 2,
  null_reps = 500,
  null_method = c("permutation", "bootstrap", "gaussian"),
  conf_level = 0.95,
  exposure_type = c("auto", "continuous")
)
```

Arguments

.data	A data frame.
.exposure	The continuous exposure column, selected with data-masking. Under the default exposure_type = "auto", check_hat_values() aborts when the exposure is detected as binary or categorical; declaring exposure_type = "continuous" passes any numeric column through the type gate.
.covariates	The covariate columns, selected with tidyselect. Numeric, logical, factor, and character columns are accepted; a factor or character column is expanded internally to one fewer column than it has levels, as treatment-contrast indicators when the factor is unordered and as polynomial contrasts when it is ordered. The two encodings span the same space, so the hat values are the same either way.
probs	A numeric vector of exposure percentiles, each in $[0, 1]$, at which candidate points are formed. Defaults to seq(0.05, 0.95, by = 0.05).

threshold	The leverage cutoff multiplier. A candidate is flagged when its hat value exceeds $\text{threshold} * p / n$. Defaults to 2.
null_reps	The number of null replicates used to calibrate $\hat{\phi}$. Defaults to 500.
null_method	The null-resampling scheme, one of "permutation" (permute observed exposures, the default), "bootstrap" (resample observed exposures with replacement), or "gaussian" (draw from a matched normal). A uniform null is deliberately not offered; see Details.
conf_level	The null quantile, strictly between 0 and 1, against which the observed $\hat{\phi}$ is compared. Defaults to 0.95.
exposure_type	One of "auto" (detect from the data, the default) or "continuous". A supplied type is authoritative and detection is not consulted, so "continuous" is rejected only when the exposure column is not numeric. Declaring it on a numeric column with few distinct values, a dose recorded at a handful of milligram levels for instance, is exactly the supported use: the unique-value heuristic reads such a column as categorical, so under "auto" the call aborts. Because numeric is the whole of the requirement, a two-valued numeric column is then accepted as well, which is the price of an authoritative declaration.

Details

The method (the paper's Box 1) fits the linear design matrix $M = [1, D, X]$, where D is the continuous exposure and X the encoded covariates, so the number of parameters is $p = q + 2$ for q encoded design columns. A numeric or logical covariate contributes one column; a factor or character covariate contributes one fewer column than it has levels, as treatment-contrast indicators when the factor is unordered and as polynomial contrasts when it is ordered. The two encodings span the same space, so q and the hat values are the same either way. For each requested exposure percentile d_q and each observed covariate row x_i , it forms the candidate point $x_* = (1, d_q, x_i)$ and computes its leverage (hat value) $h_* = x_*^T (M^T M)^{-1} x_*$. A candidate is flagged as high leverage when $h_* > \text{threshold} \times p/n$, with the default threshold of 2 reproducing the paper's $2p/n$ rule. The summary statistic $\hat{\phi}$ is the proportion of candidates that are high leverage.

Because a high value of $\hat{\phi}$ can arise from the shape of the exposure distribution alone, the diagnostic calibrates it against a null in which the exposure is drawn independently of the covariates. Each of `null_reps` replicates redraws the exposure, rebuilds the design matrix, and recomputes $\hat{\phi}$ at the same candidate points. The observed $\hat{\phi}$ is compared against the `conf_level` quantile of this null distribution; when it exceeds that quantile, `exceeds_null` is TRUE.

Three null-resampling schemes are available through `null_method`. "permutation" (the default) permutes the observed exposures, "bootstrap" resamples them with replacement, and "gaussian" draws from a normal distribution matched to the observed exposure mean and standard deviation. The permutation and bootstrap schemes are empirical: they preserve the observed exposure distribution exactly. A uniform null is deliberately not offered, because a uniform draw discards that distribution: on a skewed exposure it spreads the draws more evenly than the data are, which inflates the null quantile and costs power against a real violation.

The per-candidate `high_leverage` flag is a conservative leverage indicator that feeds $\hat{\phi}$, not a calibrated per-unit accept or reject rule. The absolute value of $\hat{\phi}$ depends steeply on the candidate percentile grid, so the null comparison, not the raw magnitude, carries the diagnostic signal.

Value

A `hat_values_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble has one row per candidate point with columns `.id` (the covariate row index), `prob` (the exposure percentile), `value` (the candidate exposure value d_q), `hat_value` (the leverage), and `high_leverage` (the logical flag). It also carries the scalar properties `@phi_hat`, `@null_dist`, `@null_quantile`, `@exceeds_null`, and `@p`, the number of model parameters $p = q + 2$ for q encoded design columns, which sets the $2p/n$ leverage cutoff.

`generics::glance()` returns a one-row tibble with `n` (the sample size), `phi_hat`, `null_quantile`, `exceeds_null` (kept logical), the counts `n_high_leverage` and `n_candidates` that `phi_hat` is the ratio of, and `p`.

References

Moodie EEM, Schulz J (2025). A Simple Diagnostic for the Positivity Assumption for Continuous Exposures. *Statistics in Medicine*, 44:e70194. doi:10.1002/sim.70194

Examples

```
set.seed(1)
n <- 200
x1 <- rnorm(n)
# The exposure depends on x1, so extreme doses are implausible for some
# covariate values.
dose <- rnorm(n, mean = x1)
df <- data.frame(dose = dose, x1 = x1)

# null_reps is small here to keep the example fast.
result <- check_hat_values(df, dose, x1, null_reps = 50)
result
```

check_hdr

*Diagnose positivity for continuous exposures with HDR non-overlap***Description**

`check_hdr()` computes the highest-density-region (HDR) non-overlap ratio of Bao and Schomaker (2025) for continuous exposures. For a common target exposure value, it reports the fraction of covariate profiles whose highest-density region excludes that value, that is, the share of the population for which the target dose is not supported.

Usage

```
check_hdr(
  .data,
  .exposure,
  .covariates,
```

```

mass = 0.95,
values = NULL,
density_estimator = hdr_density_normal(),
exposure_type = c("auto", "continuous")
)

```

Arguments

<code>.data</code>	A data frame.
<code>.exposure</code>	The continuous exposure column, selected with data-masking. Under the default <code>exposure_type = "auto"</code> , <code>check_hdr()</code> aborts when the exposure is detected as binary or categorical; declaring <code>exposure_type = "continuous"</code> accepts any numeric column that takes more than one value.
<code>.covariates</code>	The covariate columns, selected with <code>tidyselect</code> . Numeric, logical, factor, and character columns are accepted, and each reaches the density estimator's formula as it was selected. Encoding a non-numeric column is the estimator's work, which the default estimator leaves to <code>stats::lm()</code> .
<code>mass</code>	The HDR probability mass, a single value strictly between 0 and 1. Defaults to 0.95.
<code>values</code>	A numeric vector of target exposure values at which to evaluate the non-overlap ratio. <code>NULL</code> (the default) uses a 100-point grid spanning the observed exposure range.
<code>density_estimator</code>	An <code>hdr_density</code> conditional-density estimator, built with <code>hdr_density_normal()</code> (the default) or <code>new_hdr_density()</code> .
<code>exposure_type</code>	One of "auto" (detect from the data, the default) or "continuous". A supplied type is authoritative and detection is not consulted, so "continuous" is rejected only when the exposure column is not numeric. Declaring it on a numeric column with few distinct values, a dose recorded at a handful of milligram levels for instance, is exactly the supported use: the unique-value heuristic reads such a column as categorical, so under "auto" the call aborts. Because numeric is the whole of what the type asks of the column, a two-valued numeric column is then accepted as well, which is the price of an authoritative declaration. A constant column is refused whichever way the type was decided, since the ratio compares covariate profiles at a common target dose and one observed dose leaves nothing to compare.

Details

For a continuous exposure A with covariates L , fix a probability mass mass (the paper's α). At covariate profile l the HDR is the smallest set of exposure values capturing that mass of the conditional density,

$$A_\alpha(l) = \{a : f(a | l) \geq f_\alpha(l)\}, \quad P(A \in A_\alpha(l) | L = l) = \text{mass}.$$

The non-overlap ratio at a target a is the fraction of profiles whose HDR excludes it,

$$\hat{\tau}(a) = \frac{1}{n} \sum_{j=1}^n \mathbf{1}\{a \notin A_\alpha(l_j)\},$$

a value in $[0, 1]$: zero when a is supported everywhere, one when it is supported nowhere.

The conditional density is supplied by `density_estimator`. The default `hdr_density_normal()` fits `lm(exposure ~ covariates)`, treats the density as Gaussian with the residual standard deviation $\hat{\sigma}$, and uses the closed-form cutoff $\text{dnorm}(z)/\hat{\sigma}$ with $z = \Phi^{-1}((1 + \text{mass})/2)$. Membership then reduces to the interval test $|a - \hat{\mu}(l)| \leq z \hat{\sigma}$, so $\hat{\tau}(a)$ is the fraction of fitted means more than $z \hat{\sigma}$ from a .

The non-overlap ratio is a population-level, common-target quantity. It is not small merely because the density model fits well: when the exposure is strongly covariate-driven, even the best central target has a nonzero floor, because setting the whole population to one common dose is infeasible. The default normal estimator detects mean-shift support gaps, where a stratum's supported dose moves away from a target. It does not detect multimodal gaps: a hole between two modes of the true conditional density is filled by the fitted normal and reported as supported. Supply a flexible estimator through `new_hdr_density()` when multimodal structure is expected.

When the exposure is close to a deterministic function of the covariates, the fitted conditional density is narrow enough that no profile's region reaches another profile's dose, and the ratio is one everywhere inside the observed exposure range. That reading is literally correct, since a dose fixed by the covariates is supported at no other profile, but it is set by the width of the fitted density rather than by any comparison between profiles, so `check_hdr()` warns when it happens. The warning is decided on a dense grid spanning the observed range, not on values, so it reports a property of the fit and does not depend on which doses were asked about: a ratio of one at a few probes aimed into a genuine support gap is a finding rather than an artifact, and stays unremarked.

Value

An `hdr_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble has one row per target value with columns `value` (the target a) and `nonoverlap` (the ratio $\hat{\tau}(a)$). It also carries the properties `@mass` and `@density_estimator`.

`generics::glance()` returns a one-row tibble with `n` (the sample size), `mass`, `density_estimator`, `n_values` (the number of target values), and `nonoverlap_min` and `nonoverlap_max`, the range of the ratio across those targets.

References

Bao Y, Schomaker M (2025). Feasible Dose-Response Curves for Continuous Treatments Under Positivity Violations.

Examples

```
set.seed(1)
n <- 300
l <- rnorm(n)
dose <- rnorm(n, mean = 1)
df <- data.frame(dose = dose, l = l)

result <- check_hdr(df, dose, l, values = c(-2, 0, 2))
result
```

check_hdr_seq	<i>Diagnose sequential positivity for continuous exposures with HDR non-overlap</i>
---------------	---

Description

check_hdr_seq() applies the HDR non-overlap ratio of Bao and Schomaker (2025) to a time-varying continuous exposure, one time point at a time. At each time point it fits the conditional density of that exposure on the history entering its conditioning set, following the sequential-support logic of sPoRT (Chatton et al.), and reports the non-overlap ratio at the target values.

Usage

```
check_hdr_seq(
  .data,
  .exposures,
  .covariates,
  .baseline = NULL,
  mass = 0.95,
  values = NULL,
  density_estimator = hdr_density_normal(),
  lag = Inf,
  exposure_type = c("auto", "continuous")
)
```

Arguments

.data	A data frame in wide form, one row per subject.
.exposures	An ordered tidyselect of exposure columns, one per time point.
.covariates	A list of tidyselect expressions, one per time point, of the time-varying covariates. A length-one list is recycled across time points. Write it as a literal list() call, for example list(10, 11, 12), rather than a pre-built list held in a variable. Numeric, logical, factor, and character columns are accepted, and each reaches the density estimator's formula as it was selected, as in check_hdr() .
.baseline	A tidyselect of baseline covariates always included in every conditioning set, accepting the same column types as .covariates. Defaults to NULL.
mass	The HDR probability mass, a single value strictly between 0 and 1. Defaults to 0.95.
values	A numeric vector of target exposure values. NULL (the default) uses a 100-point grid spanning the pooled exposure range.
density_estimator	An <code>hdr_density</code> conditional-density estimator, built with hdr_density_normal() (the default) or new_hdr_density() .
lag	The history window: the number of earlier time points whose covariates and exposures enter each conditioning set. Defaults to Inf, the full history.

`exposure_type` One of "auto" (detect from the data, the default) or "continuous", applied to every exposure column in turn. A supplied type is authoritative and detection is not consulted, so "continuous" is rejected only when an exposure column is not numeric. Either way the error names every offending column at once, rather than the first one it meets. Declaring it on numeric columns with few distinct values, doses recorded at a handful of milligram levels for instance, is exactly the supported use: the unique-value heuristic reads such a column as categorical, so under "auto" the call aborts. Because numeric is the whole of what the type asks of a column, a two-valued numeric column is then accepted as well, which is the price of an authoritative declaration. A constant column is refused whichever way the type was decided, and the error names every constant column at once.

Details

Data are supplied in wide form, one row per subject with one exposure column per time point. For time point t the diagnostic fits the conditional density of the exposure at t on its conditioning set, then computes the point-in-time non-overlap ratio $\hat{\tau}_t(a)$ exactly as `check_hdr()` does. The conditioning set is the baseline covariates, the time-varying covariates for each time point within the lag history window, and the exposures at earlier time points within that window. With the default `lag = Inf` the full history enters.

Because each time point is treated separately, a support gap that moves a stratum's supported dose away from a target is isolated to the time point where it occurs. The default normal estimator sees mean-shift gaps only; a multimodal gap at a single time point is invisible to it, as for `check_hdr()`.

A time point whose exposure is close to a deterministic function of its conditioning set reads a ratio of one throughout that time point's observed exposure range, for the reason `check_hdr()` describes. Each time point is judged on a dense grid spanning its own range, since the shared target grid spans the pooled range and can leave a time point with a single target of its own. One warning names every such time point.

Value

An `hdr_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble has one row per time point and target value with columns `time`, `value`, and `nonoverlap`.

`generics::glance()` returns the columns `check_hdr()` reports plus `n_times`, the number of time points. The non-overlap range spans every wave rather than any one of them.

References

Bao Y, Schomaker M (2025). Feasible Dose-Response Curves for Continuous Treatments Under Positivity Violations.

Chatton A, Schomaker M, Luque-Fernandez MA, Platt RW, Schnitzer ME (2025). Is checking for sequential positivity violations getting you down? Try sPoRT!

Examples

```
set.seed(1)
n <- 200
```

```
df <- data.frame(
  l0 = rnorm(n),
  a1 = rnorm(n),
  l1 = rnorm(n),
  a2 = rnorm(n),
  l2 = rnorm(n),
  a3 = rnorm(n)
)

result <- check_hdr_seq(
  df,
  c(a1, a2, a3),
  list(l0, l1, l2),
  values = c(-2, 0, 2)
)
result
```

check_port

*Locate positivity-violating subgroups with the PoRT algorithm***Description**

check_port() runs the Positivity Regression Tree (PoRT) algorithm of Danelian et al. (2023). It grows regression trees on the covariates with the exposure as the response, then reads a fixed rule off each tree to flag subgroups in which one exposure level is rare or absent.

Usage

```
check_port(
  .data,
  .exposure,
  .covariates,
  alpha = 0.05,
  beta = 0.05,
  gamma = 2,
  n_bins = 3,
  breaks = NULL,
  exposure_type = c("auto", "binary", "categorical", "continuous"),
  ...
)
```

Arguments

.data	A data frame.
.exposure	The exposure column, selected with data-masking.
.covariates	The covariate columns, selected with tidyselect. Numeric and factor covariates are both accepted; the trees split them directly.

alpha	The minimum subgroup size for the reading rule, a fraction of the sample size strictly between 0 and 1. Defaults to 0.05.
beta	The prevalence threshold for the reading rule. A single number strictly between 0 and 0.5 (0.05 by default), or the string "gruber" for the sample-size-adaptive bound $5/(\sqrt{n} \ln n)$. The upper limit is 0.5 because the rule flags a prevalence below beta or above 1 - beta.
gamma	The maximum number of covariates that may jointly define a subgroup, a whole number of at least one. Defaults to 2.
n_bins	For a continuous exposure, the number of quantile bins into which the exposure is categorized. A whole number of at least two, 3 by default; a single bin would place every observation at one exposure level and flag every subgroup. Ignored for binary and categorical exposures.
breaks	For a continuous exposure, at least three distinct numeric cut points that override n_bins. NULL (the default) uses quantile bins.
exposure_type	One of "auto" (detect from the data, the default), "binary", "categorical", or "continuous". A supplied type is authoritative and detection is not consulted, so it is rejected only when the exposure column cannot carry it: "continuous" needs a numeric column and "binary" needs exactly two distinct values, while "categorical" asks nothing of the column. The type also selects how the exposure is read: only a continuous type is categorized into n_bins quantile bins.
...	Passed to <code>rpart::rpart.control()</code> , for example cp.

Details

PoRT searches for subgroups, defined by covariate splits, in which positivity is threatened because an exposure level is (near) unobserved. For a binary exposure the response is the indicator of the higher exposure level; for a categorical exposure the algorithm runs once per level with a one-versus-rest indicator; for a continuous exposure the exposure is first categorized into n_bins quantile bins (or the bins implied by breaks) and each bin is treated as a level. Covariates enter the trees unchanged: rpart splits numeric covariates at data-driven thresholds and groups factor levels, so no covariate categorization is performed here.

The regression trees follow the control values quoted by Danelian et al. (section 2.5): a minimum of 20 observations to attempt a split, at least six observations in a leaf, a maximum depth of 30, and no cost-complexity pruning ($cp = 0$), so that the largest number of divisions is retained. The paper states a minimum leaf size of six; rpart's own default of $\text{round}(\text{minsplit} / 3)$ is seven, so the value is pinned to six to match the paper. This affects only the smallest leaves and is immaterial at realistic sample sizes, where the alpha size gate dominates.

The reading rule (Danelian et al., section 2.3) marks a node as low support when its exposure prevalence is extreme, below beta or above 1 - beta, and its size is at least alpha of the sample. Once a node is marked its descendants are suppressed, so the reading is reported at the coarsest subgroup that isolates the violation. Because the rule is deterministic given a tree, only the tree carries stochasticity.

The combination search controls how many covariates may jointly define a subgroup (Danelian et al., section 2.4). Step one grows one tree per single covariate; covariates that produce a low-support subgroup are removed, and step k grows one tree per k-covariate combination of the remaining

covariates, up to gamma. Raising gamma uncovers joint violations that no single covariate isolates, at the cost of more trees.

Setting `beta = "gruber"` resolves the threshold to $5/(\sqrt{n} \ln n)$, the sample-size-adaptive bound used by sPoRT. It is tighter than 0.05 for samples larger than roughly 350 and looser below that, so a small sample is held to a less stringent prevalence threshold.

Value

A `port_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble has one row per reported subgroup with columns `subgroup` (the covariates defining the rule), `description` (the human-readable rule), `exposure_level` (the exposure level examined), `n` (the subgroup size), `proportion` (its share of the sample), `prevalence` (the exposure prevalence in the subgroup), and `low_support` (the logical reading-rule outcome). It also carries the `@trees`, `@alpha`, `@beta`, and `@gamma` properties.

`generics::glance()` returns a one-row tibble with `n` (the sample size), `n_subgroups` (how many subgroups were reported), `n_low_support` (how many of those the reading rule marked), and the resolved rule parameters `alpha`, `beta`, and `gamma`, which the counts cannot be read without.

References

Danelian G, Foucher Y, Léger M, Le Borgne F, Chatton A (2023). Identification of in-sample positivity violations using regression trees: the PoRT algorithm. [doi:10.1515/jci20220032](https://doi.org/10.1515/jci20220032)

Chatton A, Schomaker M, Luque-Fernandez MA, Platt RW, Schnitzer ME (2025). Is checking for sequential positivity violations getting you down? Try sPoRT!

Examples

```
set.seed(1)
n <- 400
x1 <- rnorm(n)
x2 <- factor(sample(c("a", "b"), n, replace = TRUE))
# No subject with x1 > 1 and x2 == "b" is ever treated.
ps <- ifelse(x1 > 1 & x2 == "b", 0, 0.5)
exposure <- rbinom(n, 1, ps)
df <- data.frame(exposure = exposure, x1 = x1, x2 = x2)

result <- check_port(df, exposure, c(x1, x2))
result
```

check_port_seq

Locate sequential positivity violations with the sPoRT algorithm

Description

`check_port_seq()` applies the PoRT reading rule one time point at a time along a treatment regime, the sequential Positivity Regression Tree (sPoRT) algorithm of Chatton et al. Under the stratified strategy it fits a tree at each time point among the subjects still following the regime, so that a violation masked by already-treated subjects is exposed.

Usage

```

check_port_seq(
  .data,
  .exposures,
  .covariates,
  .baseline = NULL,
  .censoring = NULL,
  strategy = c("stratified", "pooled"),
  lag = Inf,
  alpha = 0.05,
  beta = 0.05,
  gamma = 2,
  n_bins = 3,
  breaks = NULL,
  exposure_type = c("auto", "binary", "categorical", "continuous"),
  ...
)

```

Arguments

<code>.data</code>	A data frame in wide form, one row per subject.
<code>.exposures</code>	An ordered tidyselect of exposure columns, one per time point.
<code>.covariates</code>	A list of tidyselect expressions, one per time point, of the time-varying covariates. A length-one list is recycled across time points.
<code>.baseline</code>	A tidyselect of baseline covariates always included in every conditioning set. Defaults to <code>NULL</code> .
<code>.censoring</code>	An ordered tidyselect of censoring indicators, one per time point, each coded 1 when the subject is censored and 0 otherwise. When supplied, each indicator is analyzed as its own binary-exposure tree, in addition to the exposure trees. The selection must match <code>.exposures</code> in length and every indicator must be binary; monotonicity is not required of the data, since a subject censored at a time point is dropped from later censoring risk sets. An indicator that is NA at a time point is read as unknown censoring status and drops that subject from that time point's censoring risk set and every later one. Supplying <code>.censoring</code> also restricts each exposure risk set to the followers uncensored so far. Censoring is available under the stratified strategy, the only strategy currently implemented. Defaults to <code>NULL</code> , which runs the exposure trees alone.
<code>strategy</code>	The sequential strategy. Only "stratified" (per-time trees among the regime followers, the default) is implemented; "pooled" is reserved for a later release.
<code>lag</code>	The history window: the number of earlier time points whose covariates and exposures enter each conditioning set. Defaults to <code>Inf</code> , the full history.
<code>alpha, beta, gamma, n_bins, breaks</code>	As in check_port() . <code>beta</code> is resolved per time point when it is "gruber".
<code>exposure_type</code>	One of "auto" (detect from the data, the default), "binary", "categorical", or "continuous", resolved once from the exposure pooled across time points. A supplied type is authoritative and detection is not consulted, so it is rejected

only when the pooled exposure cannot carry it: "continuous" needs a numeric column and "binary" needs exactly two distinct values, while "categorical" asks nothing of the column. The type also decides the risk sets, so declaring a non-binary type on a binary exposure changes the analysis rather than only the guardrail; see Details.

... Passed to `rpart::rpart.control()`.

Details

Data are supplied in wide form, one row per subject with one exposure column per time point. Under `strategy = "stratified"` the risk set at time t is the set of subjects still following the rule. For a monotone binary treatment this is the subjects not yet initiated, that is, those whose immediately preceding treatment is the untreated level; every subject with an observed time-1 exposure is in the risk set at the first time point. Restricting to these followers is what reveals a violation that pooling over already-treated subjects would mask, because the already-treated carry their treatment forward and inflate the subgroup prevalence. A subject whose time- t exposure is missing is excluded from the time- t risk set, so n_t is always the size of the sample the tree is fitted on.

At each time point the diagnostic runs `check_port()` with the time- t exposure as the response and a conditioning set of the baseline covariates, the time-varying covariates within the lag history window, and the earlier exposures within that window. The reading rule and the combination search behave exactly as in the point diagnostic.

The exposure type and the untreated level are both resolved once from the non-missing pooled exposure across all time points, so the follower set is defined against the regime's untreated level rather than whichever levels happen to survive at a given wave, and an NA exposure, for example one a censored subject carries after leaving the study, distorts neither. The type is detected from the pooled exposure unless you declare it through `exposure_type`, and a declaration is authoritative.

Only a binary type carries the follower restriction; under any other type every subject stays in every risk set. Declaring a non-binary type on a binary exposure therefore changes the analysis rather than only the guardrail: the already-treated are pooled back in, the diagnostic stops exposing the violations that pooling over them masks, and the trees tend to flag the trivial rule that an earlier exposure predicts the current one. Declare a non-binary type on a binary exposure only when you intend to drop the regime restriction.

When a risk set at some time point is empty or too small to support the reading rule, for example a fully treated preceding wave or a single follower under the Gruber bound, that time point is skipped with a warning, contributes no rows, and records NA in `@beta`.

When `beta = "gruber"` the prevalence threshold is resolved per time point from the risk-set size n_t , as $5/(\sqrt{n_t} \ln n_t)$. Under a monotone rule the risk set shrinks over time, so the per-time threshold grows: smaller follower sets are held to a looser bound. The resolved thresholds are stored in `@beta`, ordered by time point.

Supplying `.censoring` adds a second family of per-time trees for the censoring process (Chatton et al. 2025). Each censoring indicator is analyzed as a binary exposure with the same two-sided reading rule, which flags a subgroup whose censoring prevalence is below `beta` or above `1 - beta`. The upper tail flags a subgroup nearly certain to be censored, just as the exposure trees flag a subgroup nearly certain to be treated; the lower tail flags a subgroup nearly certain to remain uncensored, the never-censored subgroups on which the counterfactual under no censoring rests (Chatton et al. 2025, page 11). Because real censoring is light in most subgroups, the lower tail usually accounts

for most of the low-support censoring rows. The censoring tree at time t runs on the subjects uncensored through the previous time point, the never-censored risk set, which is the full uncensored cohort rather than the treatment-regime followers, since the treatment rule plays no role in the censoring process. Its conditioning set is the same baseline and lag-window history the exposure trees use, with the current treatment added as a covariate. Its prevalence threshold is resolved from the size of the censoring risk set, which shrinks as earlier censoring accrues. A censoring wave in which every subject in its risk set shares the same censoring status offers no contrast to split on, so it contributes no censoring rows while still recording its resolved threshold in `@censoring_beta`. Censoring rows are marked in the results by a type column with the value "censoring", against "exposure" for the exposure trees, and this column appears only when `.censoring` is supplied.

Supplying `.censoring` also restricts the exposure risk sets. At time t the exposure risk set becomes the regime followers who are also uncensored through time $t - 1$, since a subject censored earlier has no observed exposure at time t and cannot follow the regime (Chatton et al. 2025, Figure 2). The exposure risk sets therefore shrink faster than under the follower restriction alone, and the per-time thresholds in `@beta` grow accordingly. A censored subject that carries an NA exposure after it leaves the study falls outside the restricted exposure risk sets, so its missing exposures never enter a tree.

Value

A `port_result` object, an S7 subclass of `positivity_diagnostic`. Its `@results` tibble carries the point columns plus a leading time column, and a type column distinguishing exposure from censoring rows when `.censoring` is supplied. Censoring rows report exposure_level "1", the censored level, so their prevalence is the subgroup's censoring rate. `@beta` holds the per-time exposure thresholds ordered by time point, and `@censoring_beta` holds the per-time censoring thresholds when `.censoring` is supplied.

`generics::glance()` returns the columns `check_port()` reports plus `n_times`, the number of time points. When `.censoring` is supplied it also reports `n_censoring_subgroups` and `n_censoring_low_support` beside the exposure counts, which then cover the exposure rows alone. The beta column is `NA_real_` whenever the per-time thresholds differ, as `beta = "gruber"` makes them: NA there means that the threshold varies by time point rather than that it is unknown, and the full per-time vector stays available on `@beta`. It is `NA_real_` in one further case, where the threshold genuinely is unknown: no time point resolved a threshold, because every one of them was skipped. Reading `@beta` tells the two apart, since it holds the differing values in the first case and is entirely NA in the second.

References

- Danelian G, Foucher Y, Léger M, Le Borgne F, Chatton A (2023). Identification of in-sample positivity violations using regression trees: the PoRT algorithm. doi:10.1515/jci20220032
- Chatton A, Schomaker M, Luque-Fernandez MA, Platt RW, Schnitzer ME (2025). Is checking for sequential positivity violations getting you down? Try sPoRT!

Examples

```
set.seed(1)
n <- 1000
c1 <- rnorm(n)
```

```

a1 <- rbinom(n, 1, 0.3)
# Monotone treatment: once treated, stays treated.
a2 <- a1
followers <- a1 == 0
# Subjects with c1 > 1 never initiate at time 2.
a2[followers] <- rbinom(sum(followers), 1, ifelse(c1[followers] > 1, 0, 0.4))
df <- data.frame(c1 = c1, a1 = a1, a2 = a2)

result <- check_port_seq(df, c(a1, a2), list(c1))
result

# Add censoring indicators, coded 1 when censored. Censoring is near-certain
# at time 2 where c1 > 1, which the censoring trees flag.
cens1 <- rbinom(n, 1, 0.08)
cens2 <- rbinom(n, 1, ifelse(c1 > 1, 0.98, 0.1))
df$cens1 <- cens1
df$cens2 <- cens2

check_port_seq(df, c(a1, a2), list(c1), .censoring = c(cens1, cens2))

```

check_positivity

Diagnose positivity across the applicable methods

Description

check_positivity() resolves the exposure type, detecting it from the data unless you declare one, runs the diagnostics that apply to that type with their defaults, and returns a [positivity_check](#) container. Printing it gives a report: the exposure, its type, the sample size, and the covariates stated once for the run, then one section per diagnostic reading what that diagnostic found.

Usage

```

check_positivity(
  .data,
  .exposure,
  .covariates,
  diagnostics = NULL,
  exposure_type = c("auto", "binary", "categorical", "continuous"),
  args = list()
)

```

Arguments

.data	A data frame.
.exposure	The exposure column, selected with data-masking.
.covariates	The covariate columns, selected with tidyselect. Required, with no default, so that outcome columns are not swept in by accident. The exposure column must not be selected.

diagnostics	The diagnostics to run. NULL (the default) selects the applicable set for the exposure type; otherwise a subset of "edp", "port", "hat_values", "hdr", and "extrapolation", run in the order given.
exposure_type	One of "auto" (detect from the data, the default), "binary", "categorical", or "continuous". A supplied type is authoritative: detection is not consulted, and the resolved type is forwarded to every diagnostic that runs rather than re-derived from the data by each diagnostic in turn. It must be a type the exposure column can carry, and it must apply to every entry in diagnostics; both are checked before any diagnostic runs.
args	A named list of per-diagnostic option lists, for example <code>list(port = list(alpha = 0.1))</code> . Each name must be a diagnostic being run.

Details

`check_positivity()` resolves the exposure type a single time and hands the resolved type to every diagnostic it composes, so none of them derives a type of its own and the type is announced once, through an informational message suppressed by `options(positively.quiet = TRUE)`. Every other informational message a diagnostic emits still reaches you.

Detection is a default, never an override. Under `exposure_type = "auto"` the type is inferred from the data; supply a type instead and it is authoritative, with detection not consulted at all. Either way the resolved type is checked against the exposure column before any diagnostic runs: a continuous type needs a numeric column, a binary type needs exactly two distinct values, and a categorical type asks nothing of the column. A declared type therefore aborts only when its math cannot run on the column as given, never because the unique-value heuristic read the column differently than you did. A dose recorded at eight distinct milligram levels is detected as categorical, and `exposure_type = "continuous"` runs the continuous-exposure diagnostics on it regardless.

The diagnostics that run must also apply to the resolved type. Under "auto" that type is a guess, so an inapplicable-diagnostic error names the `exposure_type` to declare in order to run the rejected diagnostics, offering only a type the exposure column can carry. Under a declared type the type is your premise, and the advice is to adjust diagnostics instead.

The default diagnostic set depends on the exposure type:

- **binary:** `check_edp()`, `check_port()`, `check_extrapolation()`
- **categorical:** `check_edp()`, `check_port()`
- **continuous:** `check_edp()`, `check_port()` (with the exposure categorized), `check_hat_values()`, `check_hdr()`

Pass diagnostics to run an explicit subset instead; the requested order is honored. Every name must be a diagnostic that `check_positivity()` composes and that applies to the resolved exposure type, otherwise the call aborts with the valid options listed.

`check_eta_bias()` and `check_density_ratios()` are never composed here. The first needs an outcome and the second needs user-supplied density ratios, so both are called directly rather than through `check_positivity()`.

Per-method options are threaded through `args`, a named list of option lists, for example `args = list(port = list(alpha = 0.1))`. Every name in `args` must be a diagnostic that is actually being run.

Value

A [positivity_check](#) object holding one [positivity_diagnostic](#) child per diagnostic, named for the diagnostic that produced it, alongside the exposure, covariates, exposure type, and sample size the call resolved.

See Also

[check_edp\(\)](#), [check_port\(\)](#), [check_hat_values\(\)](#), [check_hdr\(\)](#), and [check_extrapolation\(\)](#) for the individual diagnostics, and [check_eta_bias\(\)](#) and [check_density_ratios\(\)](#) for the two that [check_positivity\(\)](#) does not call.

Examples

```
set.seed(1)
n <- 150
x1 <- rnorm(n)
x2 <- rnorm(n)
ps <- 0.2 + 0.6 * plogis(0.5 * x1 - 0.5 * x2)
df <- data.frame(exposure = rbinom(n, 1, ps), x1 = x1, x2 = x2)

# The binary default set: edp, port, and extrapolation.
check_positivity(df, exposure, c(x1, x2))

# Run a single diagnostic with a tuned option.
check_positivity(
  df,
  exposure,
  c(x1, x2),
  diagnostics = "port",
  args = list(port = list(alpha = 0.1))
)
```

hdr_density_normal	<i>The default normal conditional-density estimator for HDR non-overlap</i>
--------------------	---

Description

[hdr_density_normal\(\)](#) builds the default conditional-density estimator used by [check_hdr\(\)](#) and [check_hdr_seq\(\)](#). It fits a linear model of the exposure on the covariates, treats the conditional density as Gaussian with the model's residual standard deviation, and supplies a closed-form HDR threshold.

Usage

```
hdr_density_normal()
```

Details

The estimator fits $\text{lm}(\text{exposure} \sim \text{covariates})$, recovering fitted means $\hat{\mu}(l)$ and a homoskedastic residual standard deviation $\hat{\sigma}$. The conditional density is $\hat{f}(a | l) = \text{dnorm}(a; \hat{\mu}(l), \hat{\sigma})$. Its HDR at mass mass is a symmetric interval around $\hat{\mu}(l)$, so the density cutoff is the closed form

$$f_{\alpha} = \text{dnorm}(z)/\hat{\sigma}, \quad z = \Phi^{-1}\left(\frac{1 + \text{mass}}{2}\right).$$

Membership of a target a in the HDR reduces to the interval test $|a - \hat{\mu}(l)| \leq z \hat{\sigma}$, and the non-overlap ratio at a is the fraction of fitted means more than $z \hat{\sigma}$ from a .

Because the working model is a single Gaussian, this estimator detects mean-shift support gaps, where a stratum's supported dose moves away from a target, but not multimodal gaps: a hole between two modes of the true conditional density is filled by the fitted normal and reported as supported. Supply a flexible estimator through `new_hdr_density()` when multimodal structure is expected.

Value

An `hdr_density` estimator object.

References

Bao Y, Schomaker M (2025). Feasible Dose-Response Curves for Continuous Treatments Under Positivity Violations.

Examples

```
estimator <- hdr_density_normal()
estimator
```

`new_hdr_density`

Construct a conditional-density estimator for the HDR diagnostic

Description

`new_hdr_density()` is the developer constructor for the pluggable conditional-density contract used by `check_hdr()` and `check_hdr_seq()`. It packages three functions into an estimator object that the HDR non-overlap ratio of Bao and Schomaker (2025) is computed against. Supply an `hdr_threshold` to give a closed-form density cutoff, or leave it `NULL` to request the numeric-grid fallback.

Usage

```
new_hdr_density(fit, density, hdr_threshold = NULL, label = "custom")
```

Arguments

<code>fit</code>	A function of (formula, data) returning fitted state as a list.
<code>density</code>	A function of (state, a, newdata) returning the conditional density $\hat{f}(a l)$ for target a over the rows of newdata.
<code>hdr_threshold</code>	An optional function of (state, newdata, mass) returning the per-row density cutoff. NULL (the default) requests the numeric-grid fallback. When it is NULL, <code>check_hdr()</code> evaluates density on a fixed grid of 1024 exposure values spanning the observed exposure range padded by its own width on each side; estimators whose conditional standard deviation is very small relative to that grid step should supply a closed-form <code>hdr_threshold</code> to keep the cutoff well resolved.
<code>label</code>	A single string naming the estimator, stored on the object and reported by the <code>check_hdr()</code> result. Defaults to "custom".

Details

The highest-density region (HDR) at a covariate profile l is the smallest set of exposure values that captures probability mass mass of the conditional density $f(a | l)$, namely $A_\alpha(l) = \{a : f(a | l) \geq f_\alpha(l)\}$. An estimator therefore needs three pieces, which map to the three arguments:

- `fit(formula, data)` fits the conditional-density model and returns any fitted state as a list. `check_hdr()` builds formula as `exposure ~ covariates` and passes the analysis data.
- `density(state, a, newdata)` returns $\hat{f}(a | l)$ for the scalar or vector target a, recycled against the rows of newdata. It receives the state from `fit()`.
- `hdr_threshold(state, newdata, mass)` returns the density cutoff $f_\alpha(l_j)$ whose HDR captures mass, one value per row of newdata. When it is NULL, `check_hdr()` instead evaluates density on a fine grid of exposure values per row and reads the cutoff off that grid, so the minimum contract is two functions.

Membership of a target a in the HDR at row j is the test $\hat{f}(a | l_j) \geq f_\alpha(l_j)$, and the non-overlap ratio $\hat{\tau}(a)$ is the fraction of rows whose HDR excludes a.

`fit()` and `density()` receive the analysis data as a data frame holding the exposure and the covariate columns as they were selected, so a factor or character covariate arrives as a factor or character column. Encoding those columns is the estimator's work. `hdr_density_normal()` leaves it to `stats::lm()`, which expands them into contrast terms from the formula; an estimator that needs a numeric design matrix should build one with `stats::model.matrix()`.

Value

An `hdr_density` estimator object.

References

Bao Y, Schomaker M (2025). Feasible Dose-Response Curves for Continuous Treatments Under Positivity Violations.

Examples

```
# A normal-equivalent estimator built through the public contract.
estimator <- new_hdr_density(
  fit = function(formula, data) {
    model <- lm(formula, data = data)
    list(model = model, sigma = sigma(model))
  },
  density = function(state, a, newdata) {
    mu <- predict(state$model, newdata = newdata)
    dnorm(a, mean = mu, sd = state$sigma)
  }
)
estimator
```

pos_violations

Binary-exposure data with planted positivity violations

Description

A simulated point-treatment dataset that carries two known positivity problems, so examples and tests can compare a diagnostic against ground truth. It holds one structural violation (a subgroup that is never exposed) and one practical near-violation (a region of covariate space where the fitted propensity approaches zero or one while both exposure levels are still observed overall). It serves [check_positivity\(\)](#) and the binary-exposure diagnostics [check_port\(\)](#), [check_edp\(\)](#), and [check_extrapolation\(\)](#).

Usage

```
pos_violations
```

Format

A tibble with 1000 rows and 4 columns:

exposure Integer treatment indicator, 0 or 1.

x1 Numeric covariate, standard normal. Its steep effect on the propensity drives the practical near-violation.

x2 Numeric covariate, standard normal. It defines the structural subgroup together with region.

region Factor covariate with levels "a" and "b".

Details

The structural violation is exact: no subject with region == "b" and x2 > 1 is exposed, because the true propensity is set to zero in that subgroup. Both exposure levels occur everywhere outside it, so the empty cell is a property of the subgroup rather than of the sample size.

The practical near-violation comes from the strong dependence of exposure on x_1 . Fitting the propensity model `glm(exposure ~ x1 + x2 + region, family = binomial())` recovers fitted values that fall below 0.01 in the lower tail of x_1 and rise above 0.99 in the upper tail, yet both exposure levels remain observed. This is the signature of a practical violation: treatment is near-deterministic in the tails without being structurally impossible.

Source

Simulated by `data-raw/make-datasets.R` in the package source repository, <https://github.com/r-causal/positively>.

pos_violations_long	<i>Longitudinal continuous-exposure data with a time-2 support gap</i>
---------------------	--

Description

A simulated wide-format longitudinal dataset with one row per subject over three time points. It carries a known support gap in the time-2 exposure, so the sequential and continuous-exposure diagnostics can be exercised against ground truth. It serves `check_hdr()`, `check_hdr_seq()`, and `check_port_seq()`.

Usage

```
pos_violations_long
```

Format

A tibble with 500 rows and 7 columns:

- id** Integer subject identifier, 1 to 500.
- i0** Numeric baseline covariate.
- a1** Numeric time-1 exposure.
- i1** Numeric time-1 covariate.
- a2** Numeric time-2 exposure, carrying the support gap.
- i2** Numeric time-2 covariate.
- a3** Numeric time-3 exposure.

Details

The exposures a_1 , a_2 , and a_3 are continuous, each taking hundreds of distinct values. The planted violation lives at time 2: a_2 is drawn on $[0, 2]$ or $[4, 6]$, so it never falls in the open interval $(2, 4)$. The time-1 and time-3 exposures are unconstrained and place values throughout that interval, which makes the gap specific to time 2 rather than a feature of the exposure distribution as a whole. The covariates and exposures follow a simple time-ordered process in which each variable depends on the one before it.

Source

Simulated by data-raw/make-datasets.R in the package source repository, <https://github.com/r-causal/positively>.

positivity_check	<i>A container for a set of positivity diagnostics</i>
------------------	--

Description

positivity_check is the S7 container returned by `check_positivity()`. It holds one `positivity_diagnostic` child per diagnostic, named for the diagnostic that produced it, together with the exposure, covariates, exposure type, sample size, and call the run resolved. Printing it gives a report: the exposure, its type, the sample size, and the covariates stated once for the run, then one section per diagnostic reading what that diagnostic found. Diagnostics with a finding to report come first, and the rest follow in the order they were requested. `sniff_violations()` is what those findings are.

Usage

```
positivity_check(
  checks = list(),
  exposure = character(0),
  exposure_type = character(0),
  covariates = character(0),
  n = integer(0),
  call = quote({
  })
)
```

Arguments

checks	A named list of <code>positivity_diagnostic</code> objects, in the order the diagnostics were requested. Each name is the diagnostic that produced the child.
exposure	The exposure column name or names, time-ordered when the diagnostics are sequential.
exposure_type	One of "binary", "categorical", or "continuous".
covariates	The covariate column names.
n	The number of observations, an integer.
call	The originating call.

Details

`summary()` gives the overview of the run: one row per statistic per child, with the columns `diagnostic`, `statistic`, `value`, and `threshold`. Each child contributes the statistics its `generics::glance()` computed, so the overview reads the same way whichever diagnostics were run.

`sniff_violations()` reports what the run found, one row per finding across every child, where `summary()` reports a statistic per child whether or not anything was found.

Extract a child diagnostic with `x$port` or `x[["port"]]`, and list the diagnostics the container holds with `names(x)`. Both extractors reject a name the container does not hold rather than returning `NULL`. A child carries its own `generics::tidy()` and `generics::glance()` methods, which are where a diagnostic's full results and its wide, typed statistics are read.

`autoplot()` draws each child's default view as one panel, and `autoplot(x, "port")` draws a single child.

Value

A `positivity_check` object. Its `summary()` method returns a `tibble` with one row per statistic per diagnostic.

Examples

```
set.seed(1)
n <- 150
x1 <- rnorm(n)
df <- data.frame(exposure = rbinom(n, 1, plogis(0.5 * x1)), x1 = x1)

res <- check_positivity(df, exposure, x1, diagnostics = "port")

# The overview across every child.
summary(res)

# What the run found, one row per finding.
sniff_violations(res)

# List the diagnostics and extract one child.
names(res)
res$port

# A child carries its own results and its own statistics.
tidy(res$port)
glance(res$port)
```

Description

`positivity_diagnostic` is the abstract S7 parent that every positively diagnostic result inherits from. It cannot be instantiated directly. It fixes the shared property set (a tidy results tibble, the exposure column names and type, the sample size, the resolved method parameters, and the originating call) and supplies the `generics::tidy()`, `generics::glance()`, `summary()`, and `print()` behavior a diagnostic inherits unless it overrides one. Package developers extend it when adding a new diagnostic.

Arguments

<code>results</code>	A tibble of tidy diagnostic output.
<code>exposure</code>	The exposure column name or names, time-ordered when the diagnostic is sequential.
<code>exposure_type</code>	One of "binary", "categorical", or "continuous".
<code>n</code>	The number of observations, an integer.
<code>params</code>	A list of method parameters as supplied and resolved.
<code>call</code>	The originating call.

Details

`generics::tidy()` returns `@results` unchanged. The inherited `generics::glance()` returns a one-row [tibble](#) holding the single column `n`, the sample size, because a subclass that adds no statistics of its own has nothing else to report. Each diagnostic overrides it to state its own statistics beside `n`, typed as they are computed.

`summary()` reports those statistics in long form, with the columns `statistic`, `value`, and `threshold`. Not every `generics::glance()` column earns a row. `value` is numeric throughout, so the character and logical statistics stay behind in the wide output, as do the sample size, the resolved method parameters, and any column that another statistic reports as its threshold, which would otherwise be stated twice. A diagnostic may override `summary()` to aggregate something else entirely: `check_extrapolation()` summarizes its per-unit results by exposure group instead.

`threshold` is the one number behind the row, stated in the units of the quantity it cuts, which are not always the units of `value`. Where the statistic is itself a reading, the cut applies to that reading, so `phi_hat` sits beside the null quantile it was compared against. Where the statistic counts how many rows crossed a cut, the cut applies to the per-row quantity rather than to the count, so a count of low-support subgroups sits beside a subgroup prevalence and a count of high-leverage candidates beside a hat value.

`threshold` is NA wherever there is no one number to state. That covers a statistic that is a raw magnitude, a statistic governed by several parameters with no single cut, a run that resolved a different cut at each wave rather than one throughout, and a statistic whose cut is real but is not one of the pairings the package tracks. An NA therefore reports that the summary has no cut to show, not that the diagnostic compared nothing against anything. The pairings are fixed inside positively; a diagnostic defined outside the package reports NA throughout.

Value

An abstract class object. Construction of a subclass returns a `positivity_diagnostic`.

pull_plot_data	<i>Pull the data behind a plot</i>
----------------	------------------------------------

Description

`pull_plot_data()` returns the tibble one view of a diagnostic result draws. A reader can rebuild the figure with `ggplot2::ggplot()` directly, take the numbers somewhere else, or read what a view is showing without rendering it.

Usage

```
pull_plot_data(x, ...)
```

Arguments

<code>x</code>	A positivity_diagnostic or a positivity_check .
<code>...</code>	Passed to methods. A class drawing more than one view takes type, the name of the view whose data to return, matched against that class's autoplot() menu. A check_port() result takes <code>low_support_only</code> as autoplot() does, and a positivity_check takes the name of the diagnostic to pull.

Details

A view returns exactly one tibble, whatever the figure does with it. Where a view splits its rows across layers, the whole frame comes back and the split stays derivable from it.

type names the view. Each method mirrors its class's [autoplot\(\)](#) menu, including the view that menu defaults to, so `pull_plot_data(x)` returns the frame `autoplot(x)` draws. Every view is built from the frame this returns, so the figure and the frame cannot disagree, and a view a result cannot draw is refused here exactly as `autoplot()` refuses it.

For a [check_edp\(\)](#) result:

- "boxplot" returns the box statistics themselves, one row per intervention, or one row per measure and intervention for the estimator variant. `ymin` and `ymax` are the fifth and the ninety-fifth percentile, `lower`, `middle`, and `upper` are the quartiles, `n` is the number of observations summarized, and `outliers` is a list-column of the values lying outside the whiskers, which the view draws as points. The estimator variant covers `edp_outcome` and `edp_treatment`; `ideal_weight` shares no scale with them and is left to the scatter view.
- "histogram", "ecdf", and "density" return the results at their own grain, one row per observation and intervention, with intervention as a factor whose levels follow the order the interventions were given in. The label is what separates the interventions rather than value, which a function intervention varies from observation to observation.
- "scatter" returns the whole estimator results tibble. The view draws the rows of finite `ideal_weight` in one layer and the infinite rows in another, and both are read off `ideal_weight`. It needs the estimator variant, and asking for it after a data-variant run is an error.

For a [check_density_ratios\(\)](#) result:

- "distribution" returns one row per ratio and time point, time and ratio. That is the grain the point-treatment histogram bins and the time-varying boxplot summarizes, and the raw ratios live nowhere else in the result.
- "cumulative" returns the cumulative-product summaries the series view draws, one row per time point and statistic, with series naming the summary in the words the key uses. It needs a time-varying input.

For a `check_eta_bias()` result:

- "bootstrap" returns one row per bootstrap draw, keyed on the estimand term, on the truncation level, and on the label its facet strip reads. The view bins the estimate. Each draw also carries the truth its term was aimed at, which the view draws as a reference line, one per term rather than one per draw, since a term's truth is the same in every draw.
- "sweep" returns one row per term and truncation level, holding the truncation the sweep is drawn against, the bias, and the lower and upper ends of the band two Monte Carlo standard errors either side of it. It needs a sweep of more than one level.

For a `check_hat_values()` result:

- "null" returns the null replicates in phi. The three numbers the figure marks beside them, null_quantile, phi_hat, and conf_level, ride as repeated columns.
- "profile" returns one row per exposure percentile the candidates were built at, prob and the fraction of that percentile's candidates reading as high leverage. The results hold one row per candidate, so the aggregation is what this view adds.

For a `check_port()` or `check_port_seq()` result there is one view, so the method takes `low_support_only` where a menu would take type. It returns one row per bar: the reported subgroups, with a leaf that more than one tree reported collapsed to a single row, the display label, the width proportional to subgroup size, and the pair of thresholds the row's prevalence was judged against in `beta_lower` and `beta_upper`.

For a `check_hdr()` or `check_hdr_seq()` result there is one view, and the frame is the results with time as a factor where a sequential run carries one, because the wave separates the curves rather than measuring anything.

For a `check_extrapolation()` result:

- "distribution" returns the results with exposure as a factor, which is what panels the histograms.
- "hull" returns the same rows with membership, the reading of `in_hull` that the stacked bars and the key both use. It needs the hull test to have run.

A `positivity_check` holds one frame per child rather than a frame of its own, so name the diagnostic to pull: `pull_plot_data(check, "port")`. The frame is that child's, unchanged, and a type passed alongside reaches it.

Value

A `tibble` holding one view's data.

Examples

```
set.seed(1)
n <- 100
x1 <- rnorm(n)
dose <- rnorm(n, mean = x1)
df <- data.frame(dose = dose, x1 = x1)
result <- check_edp(df, dose, x1, values = c(0, 1), exposure_type = "continuous")

pull_plot_data(result)
pull_plot_data(result, type = "ecdf")
```

sniff_violations	<i>Report what a set of diagnostics found</i>
------------------	---

Description

`sniff_violations()` answers what a run turned up, one row per finding, across every diagnostic it holds. It is the row-level companion to `summary()`, which reports a statistic per diagnostic whether or not anything was found.

Usage

```
sniff_violations(x, scope = c("subgroup", "overall"), ...)
```

Arguments

<code>x</code>	A positivity_diagnostic or a positivity_check .
<code>scope</code>	The kinds of row to report, any of "subgroup", "overall", and "unit". Defaults to the first two.
<code>...</code>	Passed to methods.

Details

A diagnostic reports a finding by declaring one. Each participating class states which of its statistics it reports and under which label; a class that declares nothing reports nothing, and no column is read as a flag on its behalf. A diagnostic defined outside positively therefore reports no rows until it defines a method, whatever its results hold.

`scope` names the kind of thing a row is about, and rows of different kinds are not comparable:

- "subgroup", a covariate region, which is something a reader can act on.
- "overall", a reading taken across the whole run.
- "unit", a single row of a distribution whose aggregate is the reading.

Unit rows are reachable but are never the default, because a diagnostic that reports per unit produces far more rows than one that reports per subgroup, and stacking the two buries the smaller set.

A reading contributes a row only when it fires, meaning at least one unit falls on the wrong side of it. A run in which every unit is supported reports nothing rather than reporting how well supported it was.

threshold carries the one number behind a row, stated in the units of the quantity it cuts, which are not always the units of value. Where the statistic is itself a reading, the cut applies to that reading: phi_hat sits beside the null quantile it was compared against, and a subgroup's prevalence beside the prevalence it was cut at. Where the statistic aggregates a per-row quantity, the cut applies to that quantity rather than to the aggregate, so prop_supported sits beside a Gower distance, which is what each unit was measured against, rather than beside a proportion. The number is the run's own resolved value rather than a package default, so a reader can see where a cut came from. It is NA only where no number stands behind the reading at all, as for the convex hull test, which reports containment rather than a comparison.

Value

A [tibble](#) with one row per finding and the columns diagnostic, scope, label, n, statistic, value, and threshold. value and threshold are numeric, and n is an integer count that is NA where a row is not about a set of rows.

Examples

```
set.seed(1)
n <- 300
x1 <- rnorm(n)
df <- data.frame(exposure = rbinom(n, 1, plogis(3 * x1)), x1 = x1)

res <- check_positivity(
  df,
  exposure,
  x1,
  diagnostics = c("port", "extrapolation")
)

sniff_violations(res)

# Unit rows are available but are never the default.
sniff_violations(res, scope = "unit")
```

Index

- * **datasets**
 - pos_violations, [41](#)
 - pos_violations_long, [42](#)
- [[,positively::positivity_check-method
 - (positivity_check), [43](#)
- \$,positively::positivity_check-method
 - (positivity_check), [43](#)
- autoplot(), [9](#), [44](#), [46](#)
- autoplot.density_ratios_result, [2](#)
- autoplot.edp_result, [3](#)
- autoplot.eta_bias_result, [4](#)
- autoplot.extrapolation_result, [5](#)
- autoplot.hat_values_result, [6](#)
- autoplot.hdr_result, [7](#)
- autoplot.port_result, [8](#)
- autoplot.positivity_check, [9](#)
- check_density_ratios, [10](#)
- check_density_ratios(), [2](#), [3](#), [37](#), [38](#), [46](#)
- check_edp, [12](#)
- check_edp(), [3](#), [37](#), [38](#), [41](#), [46](#)
- check_eta_bias, [15](#)
- check_eta_bias(), [4](#), [37](#), [38](#), [47](#)
- check_extrapolation, [20](#)
- check_extrapolation(), [5](#), [6](#), [37](#), [38](#), [41](#), [45](#), [47](#)
- check_hat_values, [23](#)
- check_hat_values(), [6](#), [37](#), [38](#), [47](#)
- check_hdr, [25](#)
- check_hdr(), [7](#), [28](#), [29](#), [37–40](#), [42](#), [47](#)
- check_hdr_seq, [28](#)
- check_hdr_seq(), [7](#), [38](#), [39](#), [42](#), [47](#)
- check_port, [30](#)
- check_port(), [8](#), [33–35](#), [37](#), [38](#), [41](#), [46](#), [47](#)
- check_port_seq, [32](#)
- check_port_seq(), [8](#), [42](#), [47](#)
- check_positivity, [36](#)
- check_positivity(), [9](#), [41](#), [43](#)
- generics::glance(), [11](#), [14](#), [16](#), [19](#), [22](#), [25](#), [27](#), [29](#), [32](#), [35](#), [44](#), [45](#)
- generics::tidy(), [44](#), [45](#)
- ggplot2::ggplot, [3](#), [4](#), [6–9](#)
- ggplot2::ggplot(), [46](#)
- glance(), [21](#)
- hdr_density_normal, [38](#)
- hdr_density_normal(), [26–28](#), [40](#)
- names(), [9](#)
- names,positively::positivity_check-method
 - (positivity_check), [43](#)
- new_hdr_density, [39](#)
- new_hdr_density(), [26–28](#), [39](#)
- plot.density_ratios_result
 - (autoplot.density_ratios_result), [2](#)
- plot.edp_result(autoplot.edp_result), [3](#)
- plot.eta_bias_result
 - (autoplot.eta_bias_result), [4](#)
- plot.extrapolation_result
 - (autoplot.extrapolation_result), [5](#)
- plot.hat_values_result
 - (autoplot.hat_values_result), [6](#)
- plot.hdr_result(autoplot.hdr_result), [7](#)
- plot.port_result
 - (autoplot.port_result), [8](#)
- plot.positivity_check
 - (autoplot.positivity_check), [9](#)
- pos_violations, [41](#)
- pos_violations_long, [42](#)
- positivity_check, [9](#), [36](#), [38](#), [43](#), [46–48](#)
- positivity_diagnostic, [11](#), [14](#), [19](#), [22](#), [25](#), [27](#), [29](#), [32](#), [35](#), [38](#), [43](#), [44](#), [46](#), [48](#)
- print(), [45](#)
- propensity::ipw(), [16](#)
- propensity::ps_trunc(), [19](#)

propensity::psw, [10](#)
pull_plot_data, [46](#)
pull_plot_data(), [2–8](#)

rpart::rpart.control(), [31, 34](#)

sniff_violations, [48](#)
sniff_violations(), [43, 44](#)
sniff_violations, positively::positivity_check-method
 (sniff_violations), [48](#)
stats::glm(), [18](#)
stats::lm(), [26, 40](#)
stats::model.matrix(), [40](#)
summary(), [21, 22, 44, 45, 48](#)
summary, positively::positivity_check-method
 (positivity_check), [43](#)

tibble, [44, 45, 47, 49](#)

vctrs::vec_data(), [10](#)