

Package ‘tvbounds’

September 1, 2026

Type Package

Title Sensitivity Analysis and Bounds under Total Variation Neighborhoods

Version 0.1.1

Author Filippo Palomba [aut, cre, cph]

Maintainer Filippo Palomba <fpalomba@princeton.edu>

Description Implements the sensitivity analysis framework of Palomba (2026)
``Sensitivity Analysis in Population Shares"
<<https://filippopalomba.github.io/#jmp>> for randomized experiments with
attrition, counterfactuals in structural models, and recentered
instrumental variables. Computes and plots sensitivity bounds together
with their confidence intervals and robustness summary measures. Python
module; a function-by-function port of the R package of the same name.

License MIT + file LICENSE

Encoding UTF-8

Depends Python (>= 3.9)

Imports numpy (>= 1.22),
pandas (>= 1.4),
matplotlib (>= 3.5)

Suggests juliacall,
pytest

SystemRequirements For tvbounds_counterfactual(): Julia (>= 1.9) and the
Artelys KNITRO solver with a valid license.

Contents

plot.tvbounds	2
print.tvbounds	4
print.tvbounds_summary	5
summary.tvbounds	6
tvbounds_attrition	9
tvbounds_control	14
tvbounds_counterfactual	17
tvbounds_riv	21

Index	26
--------------	-----------

plot.tvbounds

*Plot sensitivity bounds against the budget***Description**

Displays the bounds stored in a TVBounds object as functions of the budget δ of the total-variation (or contamination, or divergence) neighborhood, supplied through the `delta` column of `x.bounds` and drawn on the horizontal axis. The region between the lower bound path $\underline{\tau}(\delta)$ and the upper bound path $\overline{\tau}(\delta)$ is shaded, the outer confidence band (when the object carries one) is drawn as a lighter ribbon delimited by dashed lines, a dashed horizontal line marks the reference value τ_* , a point marks the baseline estimate at $\delta = 0$, and a dotted vertical line marks the plug-in breakdown budget $\hat{\delta}_b$ when it is interior to the budget grid.

Usage

```
x.plot(**kwargs)

tvbounds_plot(
  x,
  bands = True,
  baseline = True,
  breakdown = True,
  tau_star = 0,
  color = "#1F4E79",
  xlab = None,
  ylab = None,
  title = None,
  log_x = False,
  **kwargs
)
```

Arguments

<code>x</code>	A TVBounds object returned by <code>tvbounds_attrition()</code> , <code>tvbounds_counterfactual()</code> , or <code>tvbounds_riv()</code> .
<code>**kwargs</code>	For <code>tvbounds_plot()</code> : currently unused, accepted for compatibility with the <code>.plot()</code> method. For the <code>.plot()</code> method: further arguments forwarded to <code>tvbounds_plot()</code> .
<code>bands</code>	Logical; draw the outer confidence band when the object carries one (columns <code>ci_lower/ci_upper</code> of <code>x.bounds</code>). Sides without a band are drawn without one. Default <code>True</code> .
<code>baseline</code>	Logical; mark the baseline point estimate, that is, the value of the estimand under P_* , at $\delta = 0$. Ignored when <code>log_x = True</code> , since $\delta = 0$ cannot be placed on a logarithmic axis. Default <code>True</code> .
<code>breakdown</code>	Logical; draw a dotted vertical line, with a label, at the plug-in breakdown budget $\hat{\delta}_b$ when it is interior to the budget grid. Default <code>True</code> .
<code>tau_star</code>	Reference value τ_* of the estimand against which robustness is judged; drawn as a dashed horizontal line (default <code>0</code>).

color	Colour of the bounds, ribbons, and breakdown mark (default "#1F4E79", the paper's blue).
xlab, ylab, title	Axis labels and plot title. None (the default) uses $r"\delta"$ for the horizontal axis, the object's <code>estimand_label</code> for the vertical axis, and no title.
log_x	Logical; use a logarithmic budget axis. Rows with <code>delta <= 0</code> are dropped with a warning. Default <code>False</code> .

Details

The breakdown budget drawn by `breakdown = True` is the plug-in breakdown budget of `tvbounds_summary()`, the estimated counterpart of $\delta_b(\tau_\star) = \inf\{\delta : \underline{\tau}(\delta) \leq \tau_\star \leq \bar{\tau}(\delta)\}$: the first budget at which the bound path adjacent to $\tau_\star$ reaches it — the lower path when the baseline point estimate exceeds $\tau_\star$, the upper path otherwise. The line is annotated with the value of $\hat{\delta}_b$. No line is drawn when the breakdown budget is censored at the right endpoint of the grid, when it sits at the left endpoint, or when the baseline point estimate is missing.

Rows of `x.bounds` with missing bound values (e.g. censored or infeasible budgets) are omitted from the corresponding layer. For the counterfactual application, whose divergence budgets may exceed one, `log_x = True` switches to a logarithmic budget axis.

Value

A matplotlib Figure object, which displays in the active graphics backend and can be modified further with matplotlib calls.

References

Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.

See Also

`tvbounds_summary()` for the numerical summary measures; `.plot()` dispatches here for `TVBounds` objects.

Examples

```
import numpy as np
import pandas as pd
from tvbounds import tvbounds_attrition, tvbounds_plot

rng = np.random.default_rng(123)
n = 400
d = rng.binomial(1, 0.5, n)
s = rng.binomial(1, 1 / (1 + np.exp(-(0.5 + 0.5 * d))))
y = np.where(s == 1, 0.3 * d + rng.normal(size = n), np.nan)
dat = pd.DataFrame({"y": y, "d": d, "s": s})
fit = tvbounds_attrition(dat, outcome = "y", treatment = "d",
                        response = "s", delta = np.linspace(0, 1, 11),
                        B = 100, seed = 1)

tvbounds_plot(fit)

# Bounds only, no confidence band and no breakdown mark:
tvbounds_plot(fit, bands = False, breakdown = False)
```

print.tvbounds	<i>Print a tvbounds object</i>
----------------	--------------------------------

Description

Compact display of a TVBounds object: the application, the neighborhood over which the sensitivity bounds were computed, the sample size n , the baseline point estimate — the value of the estimand under the baseline distribution P_* , reported at $\delta = 0$ — the grid of budgets δ on which the bound paths $\underline{\tau}(\delta)$ and $\bar{\tau}(\delta)$ were evaluated, and whether inference is attached.

Usage

```
print(x)

x.print(digits = 3, **kwargs)
```

Arguments

x	A TVBounds object.
digits	Number of significant digits (default 3).
**kwargs	Further arguments; ignored.

Value

x, invisibly.

References

Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.

Examples

```
import numpy as np
import pandas as pd
from tvbounds import tvbounds_attrition

rng = np.random.default_rng(1)
n = 200
d = rng.binomial(1, 0.5, n)
s = rng.binomial(1, 1 / (1 + np.exp(-(0.4 + 0.4 * d))))
y = np.where(s == 1, 0.3 * d + rng.normal(size = n), np.nan)
fit = tvbounds_attrition(pd.DataFrame({"y": y, "d": d, "s": s}),
                        outcome = "y", treatment = "d", response = "s",
                        delta = np.linspace(0, 1, 11), bootstrap = False)

print(fit)
```

```
print.tvbounds_summary
```

Print a tvbounds summary

Description

Compact display of the summary measures computed by `tvbounds_summary()`: the plug-in and certified breakdown budgets $\delta_b(\tau_*)$ and $\hat{\delta}_b^C(\alpha)$, the shadow price of robustness $\eta(\delta) = -\underline{\tau}'(\delta)$, the robustness standard error $\varsigma_b = \sigma(\delta_b)/\eta(\delta_b)$, and the certification frontier $n^*(\delta; \bar{\alpha})$, followed by any notes on measures that could not be computed. The reference value of the estimand, τ_* , is shown as `tau_star`, matching the argument name.

Usage

```
print(x)

x.print(digits = 3, **kwargs)
```

Arguments

<code>x</code>	A TVBoundsSummary object.
<code>digits</code>	Number of significant digits (default 3).
<code>**kwargs</code>	Further arguments; ignored.

Value

`x`, invisibly.

References

Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.

Examples

```
import numpy as np
import pandas as pd
from tvbounds import tvbounds_attrition

rng = np.random.default_rng(1)
n = 200
d = rng.binomial(1, 0.5, n)
s = rng.binomial(1, 1 / (1 + np.exp(-(0.4 + 0.4 * d))))
y = np.where(s == 1, 0.3 * d + rng.normal(size = n), np.nan)
fit = tvbounds_attrition(pd.DataFrame({"y": y, "d": d, "s": s}),
                        outcome = "y", treatment = "d", response = "s",
                        delta = np.linspace(0, 1, 11), bootstrap = False)
# Without bootstrap draws the certified measures degrade to NaN, with a
# note explaining why:
print(fit.summary())
```

summary.tvbounds

Summary measures for total-variation sensitivity bounds

Description

Computes the summary measures of Palomba (2026) for a TVBounds object: the plug-in and certified breakdown budgets, the shadow price of robustness, the robustness standard error, and the certification frontier. The budget delta is the sensitivity parameter δ of the total-variation (or contamination, or divergence) neighborhood over which the bounds were computed.

Usage

```
object.summary(**kwargs)
```

```
tvbounds_summary(
    object,
    delta = None,
    tau_star = 0,
    level = 0.95,
    cost_per_unit = 50,
    jump = 0.05,
    direction = "auto"
)
```

Arguments

object	A TVBounds object returned by <code>tvbounds_attrition()</code> , <code>tvbounds_counterfactual()</code> , or <code>tvbounds_riv()</code> .
**kwargs	For the <code>.summary()</code> method: further arguments forwarded to <code>tvbounds_summary()</code> .
delta	Optional evaluation budget δ (a single nonnegative number). None (default) evaluates the measures at the plug-in breakdown budget $\delta_b(\tau_*)$, with the censoring fallback described in Details.
tau_star	Reference value τ_* of the estimand against which robustness is judged (default 0).
level	Confidence level $1 - \alpha$ for the two-sided critical value $z_{1-\alpha/2}$ used by the normal-approximation breakdown budget and the certification frontier (default 0.95, that is, <code>qnorm(0.975)</code>). The certified breakdown itself is read off the band stored in the object.
cost_per_unit	Marginal cost of one additional sampled unit, used for the cost equivalent of the certification frontier (default 50, the paper's USD benchmark from the 3ie closed-grant portfolio).
jump	Increase in the certified budget that the frontier prices, so that <code>n_star</code> is evaluated at $\delta + \text{jump}$ (default 0.05).
direction	One of "auto", "lower", "upper"; see Details.

Details

Write $\underline{\tau}(\delta)$ and $\bar{\tau}(\delta)$ for the lower and upper sensitivity bounds on the estimand at budget δ . The first is nonincreasing and the second nondecreasing in the budget, and both collapse at $\delta = 0$ to the value of the estimand under the baseline distribution P_* . Robustness is judged against a reference value τ_* of the estimand, supplied through `tau_star`, and typically zero when it is the sign of an effect rather than its magnitude that is of interest.

All measures are computed on the signed bound path adjacent to τ_* , namely $\underline{\tau}(\delta) - \tau_*$ when direction is "lower" and $\tau_* - \bar{\tau}(\delta)$ when it is "upper", so that in both cases the path starts positive when the conclusion holds at the baseline. The two directions are thus treated symmetrically, as the paper treats them by replacing the integrand g by $-g$; the displays below are written for the lower path. With direction = "auto" (the default) the lower path is used when the baseline point estimate exceeds `tau_star` and the upper path otherwise.

The breakdown budget is the smallest budget at which the bounds cease to exclude the reference value,

$$\delta_b(\tau_*) = \inf\{\delta \in [0, 1] : \underline{\tau}(\delta) \leq \tau_* \leq \bar{\tau}(\delta)\},$$

with the convention that the infimum over the empty set equals one. That convention is exactly the package's censoring rule: when the estimated path never reaches τ_* on the supplied budget grid, the breakdown is reported at the right endpoint of the grid, which is one for the total-variation and contamination neighborhoods, and is flagged by `censored = True` rather than recorded as an inequality.

Sampling uncertainty is accounted for by the certified breakdown budget,

$$\hat{\delta}_b^C(\alpha) = \inf\left\{\delta \in [0, 1] : \hat{\underline{\tau}}_n(\delta) - \frac{z_{1-\alpha/2} \hat{\sigma}_n(\delta)}{\sqrt{n}} \leq \tau_*\right\},$$

the largest budget at which the conclusion survives sampling uncertainty. Here $\hat{\underline{\tau}}_n(\delta)$ estimates the lower bound path, $\sigma(\delta)$ is the asymptotic standard deviation of that estimator in units of the estimand and $\hat{\sigma}_n(\delta)$ its estimator, and $z_{1-\alpha/2}$ is the two-sided normal critical value at the confidence level $1-\alpha$.

The shadow price of robustness is the marginal cost, in units of the estimand, of one further unit of budget, $\eta(\delta) = -\underline{\tau}'(\delta)$; it is nonnegative and nonincreasing, and the package estimates it by central differences on the budget grid. Dividing the sampling standard deviation of the bound by the shadow price converts it from units of the estimand into units of the budget, which yields the robustness standard error

$$\varsigma_b = \frac{\sigma(\delta_b)}{\eta(\delta_b)}.$$

The certification frontier is the sample size at which the population counterpart of the certified-breakdown inequality just clears the reference value at budget δ ,

$$n^*(\delta; \alpha) = \frac{z_{1-\alpha/2}^2 \sigma^2(\delta)}{(\underline{\tau}(\delta) - \tau_*)^2}.$$

It is real-valued, so the smallest certifying integer sample size is $\lfloor n^*(\delta; \alpha) \rfloor + 1$. The frontier diverges as the budget approaches the breakdown budget and is meaningful only below it: at and past the breakdown budget no sample size certifies the conclusion, and the frontier is reported as NaN. Its semi-elasticity $d \log n^*(\delta; \alpha) / d\delta$, the certification elasticity, gives the rate at which the required sample size grows with the budget; `cost_per_pp` prices a discrete version of it.

The one-row data frame measures reports the following, alongside the symbol each one corresponds to in the paper:

- `delta_b` — the plug-in breakdown budget $\delta_b(\tau_*)$, obtained as the first crossing of the estimated signed path with zero, linearly interpolated between grid points; censored flags the empty-set convention described above.
- `delta_b_ci` — the certified breakdown budget $\hat{\delta}_b^C(\alpha)$, read off the outer confidence limit stored in the object (columns `ci_lower/ci_upper` of `object.bounds`), that is, off the same band the figures draw, so that tables and figures agree on one number. NaN when the object carries no band, and `censored_ci` flags censoring as above.
- `delta_b_ci_norm` — the same certified breakdown budget in its normal-approximation form, the first crossing of $\hat{T}_n(\delta) - z_{1-\alpha/2}\hat{\sigma}_n(\delta)/\sqrt{n}$ with τ_* . It estimates the same population quantity as `delta_b_ci` and is retained as a diagnostic: the two differ only when the bootstrap distribution of the bound is asymmetric.
- `delta_eval` — the budget δ at which the local measures `eta`, `se`, `varsigma`, and `varsigma_sc` are evaluated.
- `eta` — the shadow price of robustness $\eta(\delta)$ at `delta_eval`.
- `se` — the estimated standard error of the bound in units of the estimand at `delta_eval`, that is, $\hat{\sigma}_n(\delta)/\sqrt{n}$.
- `varsigma` — the robustness standard error $\varsigma_b = \sigma(\delta_b)/\eta(\delta_b)$, computed as `se * sqrt(n) / eta` and therefore expressed in units of the budget rather than in units of the estimand.
- `varsigma_sc` — the finite-sample analogue `se / eta`, equal to $\varsigma_b/\sqrt{n}$, which is the sampling standard deviation of the breakdown budget itself at the realized sample size.
- `frontier_at`, `n_cur`, `n_star`, `delta_n`, `cost_per_pp` — the certification frontier. `frontier_at` is the budget at which the frontier is priced and `n_cur` the real-valued frontier $n^*(\delta; \alpha)$ there, while `n_star` is the smallest certifying integer sample size $\lfloor n^*(\delta; \alpha) \rfloor + 1$ at the budget raised by `jump`. `delta_n` is that sample size net of the realized `n`, and `cost_per_pp` the implied cost of raising the certified budget by one percentage point, at `cost_per_unit` per sampled unit.
- `label`, `direction`, `n`, `point`, `tau_star` — the estimand label, the resolved direction, the sample size n , the baseline estimate of the estimand at $\delta = 0$, and the reference value τ_* .

The evaluation budget for `eta`, `se`, and `varsigma` is the `delta` argument when supplied; when `delta = None` (the default) it is the plug-in breakdown budget, replaced by the certified breakdown budget when the plug-in breakdown is censored, as in the paper. The frontier is priced at the certified breakdown (default) or at the supplied `delta`.

Objects without inference (the recentered-IV and counterfactual applications carry none by design) degrade gracefully: the plug-in measures are reported, the certified and frontier measures are NaN, and the `notes` field explains why.

Value

An object of class `TVBoundsSummary`: an object with a one-row data frame measures (columns `label`, `direction`, `n`, `point`, `tau_star`, `delta_b`, `censored`, `delta_b_ci`, `censored_ci`, `delta_b_ci_norm`, `delta_eval`, `eta`, `se`, `varsigma`, `varsigma_sc`, `frontier_at`, `n_cur`, `n_star`, `delta_n`, `cost_per_pp`) and metadata fields (`application`, `estimand_label`, `neighborhood`, `divergence`, `direction`, `tau_star`, `delta_eval`, `level`, `band_level`, `zc`, `cost_per_unit`, `jump`, `has_se`, `has_ci`, `notes`, `call`). `Details` gives the symbol of the paper each column of measures corresponds to. Printed compactly by `print()`.

References

Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.

See Also

`tvbounds_plot()` to display the bounds; `.summary()` dispatches here for TVBounds objects.

Examples

```
import numpy as np
import pandas as pd
from tvbounds import tvbounds_attrition, tvbounds_summary

rng = np.random.default_rng(123)
n = 400
d = rng.binomial(1, 0.5, n)
s = rng.binomial(1, 1 / (1 + np.exp(-(0.5 + 0.5 * d))))
y = np.where(s == 1, 0.3 * d + rng.normal(size = n), np.nan)
dat = pd.DataFrame({"y": y, "d": d, "s": s})
fit = tvbounds_attrition(dat, outcome = "y", treatment = "d",
                        response = "s", delta = np.linspace(0, 1, 11),
                        B = 100, seed = 1)

tvbounds_summary(fit)

# Evaluate the measures at a chosen budget instead of the breakdown:
tvbounds_summary(fit, delta = 0.2)
```

tvbounds_attrition	<i>Sensitivity bounds for randomized experiments with attrition</i>
--------------------	---

Description

Computes sensitivity bounds on the average treatment effect for the always-observed subpopulation of a randomized experiment with attrition, following Palomba (2026). Writing P_0 for the distribution of the data, the estimand is

$$\tau_0 := \mathbb{E}_{P_0}[Y(1) - Y(0) \mid S(0) = 1, S(1) = 1],$$

the average treatment effect on the units that respond under either arm. The bounds are indexed by a budget $\delta \in [0, 1]$, supplied through `delta`, which caps the total variation distance between the outcome distribution of the compliers (units that respond only under treatment) and that of the always-observed units, $\text{TV}(P_C \parallel P_{AO}) \leq \delta$. At $\delta = 0$ the two distributions coincide and the bounds collapse to the baseline difference in means among respondents (the estimand under missingness completely at random); at $\delta = 1$ the restriction is vacuous and they equal the trimming bounds of Lee (2009). Optionally computes a nonparametric bootstrap (with clustering) for standard errors and a percentile confidence band, and covariate-pooled bounds that allocate a single budget optimally across covariate cells.

Usage

```
tvbounds_attrition(
    data,
    outcome,
    treatment,
    response,
    covariates = None,
```

```

delta = np.linspace(0, 1, 101),
neighborhood = "tv",
bootstrap = True,
B = 1000,
cluster = None,
level = 0.95,
min_obs = 5,
seed = None,
verbose = False
)

```

Arguments

data	A data frame containing the columns named by outcome, treatment, response, and (optionally) covariates and cluster.
outcome	String; name of the numeric column holding the outcome Y . May be NaN for non-respondents (and for the occasional respondent with item non-response, which is dropped from the outcome samples).
treatment	String; name of the binary 0/1 column holding the treatment D (1 = treated).
response	String; name of the binary 0/1 column holding the response indicator S (1 = outcome observed).
covariates	Optional list of column names to stratify on, forming the discrete covariate X ; cells are formed by their interaction. The covariate columns must be free of missing values — recode missing values into an explicit category first. Default None (no stratification).
delta	Numeric vector of budget values $\delta \in [0, 1]$ at which the bounds are evaluated; sorted and de-duplicated internally. The baseline ($\delta = 0$) and Lee ($\delta = 1$) end-points are always computed internally even when absent from delta. Default <code>np.linspace(0, 1, 101)</code> .
neighborhood	Either "tv" (total variation, default) or "contamination"; see the Neighborhoods section.
bootstrap	Logical; compute bootstrap standard errors and the percentile confidence band. Default True.
B	Number of bootstrap replications B . Default 1000.
cluster	Optional string; name of a cluster identifier column. When supplied, the bootstrap resamples whole clusters with replacement. Default None (units resampled independently).
level	Confidence level of the percentile band, $1 - \alpha$ in the notation of the Inference section. Default 0.95.
min_obs	Minimum number of observed outcomes per arm for a covariate cell to be retained. Default 5.
seed	Optional integer seed for the bootstrap. When non-None, the random-number-generator state is set locally and restored on exit, so the call has no side effect on the caller's RNG. Default None.
verbose	Logical; emit progress messages. Default False.

Value

An object of class TVBounds: an object with attributes

- application: "attrition".
- bounds: data frame with one row per requested budget and columns delta, lower, upper, holding the sensitivity bounds $\underline{\tau}(\delta)$ and $\bar{\tau}(\delta)$ (or their pooled covariate counterparts $\underline{\tau}_X(\delta)$ and $\bar{\tau}_X(\delta)$ when covariates is supplied), plus, when bootstrap = True, lower_se, upper_se, ci_lower, ci_upper (outer percentile band endpoints at level).
- point: the estimate of τ_0 at $\delta = 0$, where the bounds collapse to the difference in means among respondents $\tau_{\text{MCAR}}(P_0)$ (with covariates, to its covariate-weighted analogue).
- n: number of rows of data used.
- neighborhood, level, B, estimand_label, call as documented in the package overview (level and B are NaN without inference).
- details: dict with p_star (the complier share π , which also sets the trimming mass), response_rate (the arm-specific response rates $(\hat{r}_0, \hat{r}_1)$) and n_by_arm / n_respondents_by_arm (named, control and treated), lee (dict with the Lee-endpoint lower / upper, $\underline{\tau}_{\text{Lee}}$ and $\bar{\tau}_{\text{Lee}}$, of the estimated specification), lee_nocov (when covariates are used), breakdown (dict with the point-estimate breakdown budget δ_b at the reference value $\tau_\star = 0$ and, with inference, the confidence-band breakdown budget; NaN when censored beyond one), pooled (with covariates: per-stratum table, coverage, dropped cells, and the within-stratum reference curve pw, $\underline{\tau}_X^{\text{pw}}(\delta)$ and $\bar{\tau}_X^{\text{pw}}(\delta)$, under total variation), n_clusters (with cluster), and boot (replicate counts, width standard errors, and — when memory-reasonable — the matrices of bound draws).

Setup and estimand

Let $D \in \{0, 1\}$ be the binary treatment and, for $d \in \{0, 1\}$, let $Y(d)$ be the potential outcome and $S(d) \in \{0, 1\}$ the potential response indicator. Their realized counterparts are

$$Y = DY(1) + (1 - D)Y(0), \quad S = DS(1) + (1 - D)S(0),$$

so that the observed data are (YS, S, D) : the outcome is recorded only when $S = 1$. The response pattern $(S(0), S(1))$ partitions the population into always-observed units ($S(0) = 1, S(1) = 1$), never-observed units ($S(0) = 0, S(1) = 0$), compliers ($S(0) = 0, S(1) = 1$) and defiers ($S(0) = 1, S(1) = 0$); the estimand τ_0 is the average treatment effect on the first of these groups.

Two assumptions are maintained. Random assignment enters as $(S(0), S(1)) \perp\!\!\!\perp (Y(0), Y(1))$, labelled (MCAR), and the monotonicity condition of Lee (2009), $S(1) \geq S(0)$ almost surely, labelled (Mono) — treatment never causes a unit that would respond under control to attrit — rules out defiers. Under (Mono) the outcome distribution of the observed treated, P_T , is a mixture of the complier and always-observed outcome distributions,

$$P_T = \pi P_C + (1 - \pi) P_{\text{AO}}, \quad \pi = 1 - \frac{r_0}{r_1},$$

where $r_1 := P_0[S = 1 \mid D = 1]$ and $r_0 := P_0[S = 1 \mid D = 0]$ are the arm-specific response rates and π is the complier share. The always-observed control mean is identified, $\mu^{\text{AO}}(0) = \mathbb{E}_{P_0}[Y \mid D = 0, S = 1]$, whereas the always-observed treated mean $\mu^{\text{AO}}(1)$ is only partially identified; the bounds on τ_0 follow by subtracting the identified control mean.

The complier share is estimated by $\hat{\pi} = \max\{1 - \hat{r}_0/\hat{r}_1, 0\}$ and returned as details["p_star"], and the estimated response rates $(\hat{r}_0, \hat{r}_1)$ as details["response_rate"]. A negative unconstrained estimate of π is sampling noise under (Mono) and is projected to zero, in which case the bounds collapse to the baseline difference in means at every budget.

Neighborhoods

Two robustness sets are available through neighborhood. Both are indexed by the budget δ and both restrict the unobserved complier outcome distribution P_C relative to the unobserved always-observed outcome distribution P_{AO} :

- "tv" (default): the total variation neighborhood of the paper, $TV(P_C \parallel P_{AO}) \leq \delta$, so the two distributions may disagree on at most a δ fraction of their mass.
- "contamination": a one-sided strengthening in which P_C lies in the contamination neighborhood of P_{AO} (Huber 1964), that is $P_C = (1 - \delta)P_{AO} + \delta R$ for some distribution R , equivalently $P_C \geq (1 - \delta)P_{AO}$ as measures: a $(1 - \delta)$ -share of the compliers has outcomes distributed exactly like the always-observed units, and only the remaining δ -share may differ arbitrarily.

Under total variation the rescaling identity $TV(P_C \parallel P_T) = (1 - \pi) TV(P_C \parallel P_{AO})$ recenters the restriction on the identified distribution P_T , so that the candidate complier distributions Q , among which P_C lies, range over the robustness set

$$\mathcal{Q}_C(\delta) := \{Q \in \Delta(\mathcal{Y}) : TV(Q \parallel P_T) \leq (1 - \pi)\delta, \pi Q \leq P_T\},$$

where $\Delta(\mathcal{Y})$ denotes the distributions on the outcome space and the second restriction is the mixture structure of the observed treated arm. The resulting sensitivity bounds $\underline{\tau}(\delta)$ and $\bar{\tau}(\delta)$ on τ_0 are available in closed form as trimmed means of P_T . Writing $F_{P_T}^{-1}$ for the quantile function of the observed treated outcomes and

$$s_L(\delta) := F_{P_T}^{-1}(\pi\delta), \quad s_U(\delta) := F_{P_T}^{-1}(1 - (1 - \pi)\delta),$$

the upper bound is

$$\bar{\tau}(\delta) = \mathbb{E}_{P_T}[Y \mathbf{1}\{s_L(\delta) < Y < s_U(\delta)\}] + \frac{1}{1 - \pi} \mathbb{E}_{P_T}[Y \mathbf{1}\{Y \geq s_U(\delta)\}] - \mu^{AO}(0),$$

and the lower bound is obtained symmetrically, trimming at $t_L(\delta) := F_{P_T}^{-1}((1 - \pi)\delta)$ and $t_U(\delta) := F_{P_T}^{-1}(1 - \pi\delta)$.

Under contamination, combining the same mixture identity with $P_C \geq (1 - \delta)P_{AO}$ pins the density $w := dP_C/dP_T$ between $(1 - \delta)/(1 - \delta\pi)$ and $1/\pi$, and the bounds are again trimmed means of P_T , now with effective trimming mass $\delta\pi$. The contamination neighborhood is contained in the total variation one at every budget, so its bounds are weakly tighter, and the two families share the same endpoints: the baseline at $\delta = 0$ and, at $\delta = 1$, the Lee (2009) bounds $\underline{\tau}_{Lee} = \mathbb{E}_{P_T}[Y \mid Y \leq y_{1-\pi}] - \mu^{AO}(0)$ and $\bar{\tau}_{Lee} = \mathbb{E}_{P_T}[Y \mid Y \geq y_\pi] - \mu^{AO}(0)$, where $y_u := F_{P_T}^{-1}(u)$.

Both bound families are monotone in the budget by construction: the robustness sets are nested in δ .

Covariates

When covariates is supplied, units are stratified on the interaction of the covariate columns, which plays the role of a discrete covariate X with support $\mathcal{X}$. Cells with fewer than `min_obs` observed outcomes in either arm are dropped with a warning, and the retained cells are weighted by their control-respondent shares, which under (Mono) are the covariate distribution of the always-observed population, $P_{X|D=0,S=1} = P_{X|AO}$. Within a cell the complier share $\pi(x)$ and the observed treated outcome distribution $P_T(x)$ are identified, and the cell-level construction is the one above.

Two ways of spending the budget across cells are distinguished. The within-stratum ("pointwise") restriction imposes $\text{TV}(P_C(x) \parallel P_{AO}(x)) \leq \delta$ in every cell separately, giving one robustness set $\mathcal{Q}_C^{\text{pw}}(\delta; x)$ per cell, whereas the pooled restriction caps only the average departure,

$$\int_{\mathcal{X}} \text{TV}(P_C(x) \parallel P_{AO}(x)) dP_{X|AO}(x) \leq \delta,$$

and so allows heterogeneity across cells inside the single robustness set $\mathcal{Q}_{C,X}(\delta)$. The pooled restriction is the weaker of the two, so $\underline{\tau}_X(\delta) \leq \underline{\tau}_X^{\text{pw}}(\delta)$ and $\bar{\tau}_X^{\text{pw}}(\delta) \leq \bar{\tau}_X(\delta)$, with equality at $\delta = 0$ and at $\delta = 1$, where both collapse to the covariate Lee (2009) bounds $\underline{\tau}_{\text{Lee},X}$ and $\bar{\tau}_{\text{Lee},X}$.

For neighborhood = "tv" the reported bounds are the pooled (joint) bounds $\underline{\tau}_X(\delta)$ and $\bar{\tau}_X(\delta)$: the budget allocation $t : \mathcal{X} \rightarrow \mathbb{R}_+$ subject to $\mathbb{E}_{P_T}[t(X)] \leq (1 - \pi)\delta$ is solved exactly by a greedy fill over the stratum value functions $V(t; x)$, which are piecewise linear in the cell budget and concave for the upper bound and convex for the lower one. The within-stratum bounds $\underline{\tau}_X^{\text{pw}}(\delta)$ and $\bar{\tau}_X^{\text{pw}}(\delta)$ are returned in details["pooled"]["pw"] for reference. For neighborhood = "contamination" the reported bounds impose the common budget delta within every retained cell and aggregate; they remain weakly inside the total variation bounds at every budget.

Inference

The bootstrap resamples the full observation $W = (Y, S, D)$, together with the covariates, with replacement — whole clusters when cluster is supplied — and recomputes the entire bounds curve on each replicate, yielding $\hat{\tau}^{(b)}(\delta)$, $b = 1, \dots, B$. Each replicate therefore redraws the arm-specific response rates and hence $\hat{\pi}$, so the reported standard errors carry the estimation uncertainty in π , which the naive variance $\hat{\sigma}_{\text{naive}}^2(\delta)$ omits by treating $\hat{\pi}$ as known; in the paper's influence function $\psi_{\text{full}}(W; \delta)$ this uncertainty is the term $\kappa(\delta)\psi_{\pi}(W)$. The reported upper_se is the standard deviation of the draws $\hat{\tau}^{(b)}(\delta)$ across replicates, that is the paper's $\hat{\sigma}_{\text{boot}}(\delta)$ divided by $\sqrt{n}$ for a sample of size n , and lower_se is its counterpart for the lower bound. The reported confidence band is the percentile band: writing α for the value of 1 - level, ci_lower is the $\alpha/2$ quantile of the lower-bound draws and ci_upper the $1 - \alpha/2$ quantile of the upper-bound draws, the outer envelope of the identified set. Replicates on which the bounds cannot be computed are dropped and counted (a warning reports their number).

References

- Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.
- Lee, D. S. (2009). "Training, Wages, and Sample Selection: Estimating Sharp Bounds on Treatment Effects." *Review of Economic Studies*, 76(3), 1071-1102.
- Huber, P. J. (1964). "Robust Estimation of a Location Parameter." *Annals of Mathematical Statistics*, 35(1), 73-101.

See Also

[tvbounds_plot\(\)](#) and [tvbounds_summary\(\)](#) for reporting.

Examples

```
import numpy as np
import pandas as pd
from tvbounds import tvbounds_attrition

rng = np.random.default_rng(123)
n = 400
```

```

d = rng.binomial(1, 0.5, n)
s = rng.binomial(1, np.where(d == 1, 0.9, 0.7))
y = np.where(s == 1, rng.normal(loc = 0.3 * d), np.nan)
x = rng.binomial(1, 0.5, n)
dat = pd.DataFrame({"y": y, "d": d, "s": s, "x": x})

## Total variation bounds with a small bootstrap
fit = tvbounds_attrition(dat, outcome = "y", treatment = "d",
    response = "s", delta = np.linspace(0, 1, 11), B = 50, seed = 1)
fit.bounds
fit.details["lee"]

## Contamination neighborhood, no inference: weakly tighter bounds
fit_c = tvbounds_attrition(dat, outcome = "y", treatment = "d",
    response = "s", delta = np.linspace(0, 1, 11),
    neighborhood = "contamination", bootstrap = False)
(fit_c.bounds["lower"] >= fit.bounds["lower"] - 1e-12).all()

## Covariate-pooled bounds
fit_x = tvbounds_attrition(dat, outcome = "y", treatment = "d",
    response = "s", covariates = "x", delta = np.linspace(0, 1, 11),
    bootstrap = False)

```

tvbounds_control

Control options for the counterfactual solver

Description

Constructs the list of tuning options consumed by `tvbounds_counterfactual()`. The defaults reproduce the settings used for the counterfactual application in Palomba (2026).

Usage

```

tvbounds_control(
    maxsolves = 10,
    startptrange = 0.01,
    use_optim = False,
    time_limit = 60,
    iterations = 100,
    outer_iterations = 3,
    inner_opt = None,
    outer_opt = None,
    knitro_options = {},
    eta_min = 1e-120,
    lower_limit = -10,
    psi_tv_eps = 1e-04,
    tvac_tau = 0.001,
    tvmix_tau = 0.001,
    purekl_acap = 500,
    tvmix_kappa = None
)

```

Arguments

maxsolves	Integer, number of multi-start restarts of the outer optimization over the structural parameter θ for each budget and side.
startptrange	Positive scalar; restarts after the first perturb the initial θ uniformly on $[-\text{startptrange}, \text{startptrange}]$ coordinate-wise (clipped to the parameter box).
use_optim	Logical; if True the outer optimization uses Optim.jl (projected L-BFGS with the analytic envelope-theorem gradient) instead of a nested KNITRO solve. The Optim fallback avoids nested KNITRO contexts, which segfault with some KNITRO.jl versions.
time_limit	Positive scalar, wall-clock limit in seconds per outer Optim run (used only when <code>use_optim = True</code>).
iterations	Positive integer, inner iteration limit per outer Optim run (used only when <code>use_optim = True</code>).
outer_iterations	Positive integer, number of Fminbox outer iterations per Optim run (used only when <code>use_optim = True</code>).
inner_opt	Path to a KNITRO option file for the inner (dual) problem, or None for the shipped default <code>inner.opt</code> .
outer_opt	Path to a KNITRO option file for the outer problem (over the structural parameter θ), or None for the shipped default <code>outer.opt</code> . The package also ships <code>outer_fast.opt</code> (analytic envelope gradient, looser tolerances, suited to plotting grids) and <code>outer_boot_tv.opt</code> ; point this argument at <code>tvbounds.julia_file("opt", "outer_fast.opt")</code> to use them.
knitro_options	Dict of individual KNITRO options (e.g. <code>dict(maxit = 500, outlev = 2)</code>). These are merged into <i>both</i> the inner and the outer option files by writing merged copies to the temporary directory (<code>tempfile.gettempdir()</code>); entries override options already present and are appended otherwise. Values must be scalar strings, numbers, or booleans (booleans are written as 0/1).
eta_min	Positive scalar, lower bound on the dual variable η , the multiplier pricing the divergence budget (kept strictly positive so the perspective function in the dual objective is well defined).
lower_limit	Scalar; inner (dual) objective values at or below this threshold are treated as unbounded below (the inner solver's infeasibility guard).
psi_tv_eps	Positive scalar, Huber smoothing scale for the kinked total-variation conjugate ϕ_{TV}^* . The smoothed conjugate lies above the exact one, so computed bounds remain outward-conservative.
tvac_tau	Positive scalar, soft-max temperature for the log-sum-exp term of the "TVac" divergence, that is, total variation restricted to distributions with $P \ll P_*$.
tvmix_tau	Positive scalar, soft-max temperature for the log-sum-exp term of the "TVmix" divergence, that is, total variation intersected with the mixture constraint $P \geq \kappa P_*$.
purekl_acap	Positive scalar, overflow clamp on the exponent of the conjugate of the pure Kullback-Leibler entropy $\phi_{\text{KL}}(s) = s \log s - s + 1$, which increases exponentially in its argument.
tvmix_kappa	None or a scalar in $[0, 1]$: the contamination weight κ of the mixture constraint $P \geq \kappa P_*$ used by the "TVmix" and "TVmixC" divergences (the perturbed distribution must contain the baseline as a mixing component with weight κ). None

ties κ to the budget as $\kappa = 1 - \delta$, matching Palomba (2026). An explicit value decouples the two parameters; the reduced "TVmix" program requires $\kappa \geq 1 - \delta$ (so that the total-variation constraint is redundant), while "TVmixC" supports any κ in $[0, 1)$.

Details

The options are stated in the notation of the paper. A candidate distribution P is measured against the baseline P_* by the divergence $D_\phi(P||P_*)$ generated by an entropy function ϕ , and the budget δ caps it. At a fixed structural parameter θ the inner problem is solved in its dual form, over the multipliers (ζ, η, λ) attached respectively to the total-mass constraint, to the divergence budget – so that η is the shadow price of robustness – and to the moment restrictions. The dual objective integrates against the baseline the perspective of the convex conjugate ϕ^* ,

$$(\phi^*)^\pi(g(U; \theta) - \lambda^\top m(U; \theta) - \zeta, \eta),$$

with g the counterfactual criterion and m the moment function. Under total variation, $\phi_{TV}(s) = |s - 1|/2$, the conjugate ϕ_{TV}^* is piecewise linear and the perspective collapses to the kinked $\max\{g(U; \theta) - \lambda^\top m(U; \theta) - \zeta, -\eta/2\}$; the smoothing options below round those kinks off, always from above, so the computed bounds stay outward-conservative. The contamination weight κ of the mixture constraint $P \geq \kappa P_*$ is set by `tvmix_kappa`.

Value

A dict of class `TVBoundsControl` with the (validated) options above.

References

- Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.
- Christensen, T. and B. Connault (2023). "Counterfactual Sensitivity and Robustness." *Econometrica*, 91(1), 263-298.

See Also

[tvbounds_counterfactual\(\)](#)

Examples

```
from tvbounds import tvbounds_control

ctrl = tvbounds_control()
ctrl["maxsolves"]

# a faster configuration for exploratory grids
tvbounds_control(maxsolves = 3, knitro_options = dict(maxit = 200))
```

tvbounds_counterfactual

Sensitivity bounds for counterfactual predictions in structural models

Description

Computes the lower and upper sensitivity bounds $\underline{k}(\delta)$ and $\bar{k}(\delta)$ on a counterfactual $\mathbb{E}_P[g(U; \theta)]$, when the distribution P of the latent variables U ranges over a divergence neighborhood of the simulated baseline P_* with budget δ , and the structural parameter θ ranges over the values compatible with the moment conditions $\mathbb{E}_P[m(U; \theta)] \in \mathcal{M}(\rho)$, following Palomba (2026) and Christensen and Connault (2023). This is the only function in the package that supports general ϕ -divergences beyond total variation (spelled out at first use; "TV" below) and contamination.

Usage

```
tvbounds_counterfactual(
  moments,
  d,
  theta_lb,
  theta_ub,
  delta,
  divergence = "KL_chi2",
  side = "both",
  U = None,
  M = 50000,
  u_dim = None,
  gamma = None,
  gradient = None,
  theta_init = None,
  control = tvbounds_control(),
  seed = None,
  verbose = True
)
```

Arguments

moments	The moment/counterfactual function: a length-2 sequence of strings (file, fname), a single string naming a Julia function, or a Python function. See Details.
d	Integer, the number of moment conditions d_m (columns of G).
theta_lb, theta_ub	Numeric vectors of equal length: the box for the structural parameter θ over which the outer problems optimize. Their common length fixes the dimension of θ .
delta	Numeric vector of strictly positive budgets δ (the sensitivity parameter of the neighborhood), without duplicates. For the total-variation family the budget must lie in $(0, 1]$; for "KL_chi2", "KL", and "chi2" any positive value is allowed.
divergence	Divergence keyword; one of "KL_chi2" (default), "KL", "chi2", "TV", "TVmix", "TVmixC", "TVac". See Details.

side	"both" (default), "lower", or "upper": which bound problems to solve at each budget.
U	Optional $M \times u_dim$ numeric matrix of latent draws $U^{(1)}, \dots, U^{(M)}$, one per row. When None, scrambled-Halton uniforms are generated (see Details) and M , u_dim are required; when supplied, M and u_dim are taken from its dimensions.
M	Integer, the number M of simulated draws when $U = \text{None}$ (default 50000, the setting of the paper).
u_dim	Integer, the dimension d_z of the latent draw when $U = \text{None}$ (at most 15).
gamma	Optional dict: an arbitrary payload forwarded to the moments function (as <code>obj.gamma</code> for Julia moments, as the third argument for Python moments).
gradient	How to differentiate the moments with respect to θ in the outer optimization: None (default; automatic differentiation for Julia moments, finite differences for Python moments), the string "fd" (finite differences), the name of a Julia function <code>jac(theta, U, obj)</code> , or a Python function <code>f(theta, U, gamma)</code> . A user-supplied gradient must return either the stacked $(M \times (d+1)) \times 1$ Jacobian of (K, G) (K rows first, then G in column-major order) or a dict with components K ($M \times 1$) and G ($M \times d \times 1$), where $1 = \text{len}(\text{theta_lb})$.
theta_init	Optional numeric vector, the initial structural parameter θ for the outer optimization (defaults to the midpoint of the box). Set it to the baseline estimate of the model: the reported baseline point is the plug-in counterfactual $k(\theta; P_*) = \mathbb{E}_{P_*}[g(U; \theta)]$ at <code>theta_init</code> , and is only returned when <code>theta_init</code> is supplied.
control	A dict created by <code>tvbounds_control()</code> with solver tuning options.
seed	Optional integer seed for the scrambled-Halton draws.
verbose	Logical; print solver progress (default True).

Details

Setup. Let $U \in \mathcal{Z}$ collect the latent variables of the structural model (taste shocks, unobserved heterogeneity, measurement errors) and let P_* be the baseline distribution the econometrician postulates for them. At a structural parameter $\theta \in \Theta$, a distribution P is compatible with the model when $\mathbb{E}_P[m(U; \theta)] \in \mathcal{M}(\rho)$, with m the moment function and $\mathcal{M}(\rho)$ the moment constraint set; the object of interest is the counterfactual $\mathbb{E}_P[g(U; \theta)]$, the expectation of a known criterion g that is linear in P at fixed θ . The robustness set collects the distributions that are compatible with the model and within budget of the baseline,

$$\mathcal{P}_\phi(\theta; \rho, P_*, \delta) := \{P : D_\phi(P \| P_*) \leq \delta, \mathbb{E}_P[m(U; \theta)] \in \mathcal{M}(\rho)\},$$

where $D_\phi(P \| P_*)$ is the ϕ -divergence selected by divergence and δ the budget, and the reported bounds are the nested extrema

$$\underline{k}(\delta) = \inf_{\theta \in \Theta} \inf_{P \in \mathcal{P}_\phi(\theta; \rho, P_*, \delta)} \mathbb{E}_P[g(U; \theta)], \quad \bar{k}(\delta) = \sup_{\theta \in \Theta} \sup_{P \in \mathcal{P}_\phi(\theta; \rho, P_*, \delta)} \mathbb{E}_P[g(U; \theta)].$$

The optimization over P at a fixed θ is the inner problem, solved in its dual form; the optimization over θ is the outer problem.

Correspondence between the paper and the code. The paper writes the counterfactual integrand as g and the moment function as m . The interfaces below fill an array named `K` with the values of g and an array named `G` with the values of m : read `K` as g and `G` as m throughout. The solver takes the moment conditions in centered equality form, $\mathbb{E}_P[m(U; \theta)] = 0$, so a nonzero target ρ is absorbed by centering the moment function. The argument `d` is the number of moment conditions, d_m ; the

common length of `theta_lb` and `theta_ub` is the dimension of θ ; and `u_dim` is the dimension d_z of a single latent draw.

Moments specification. moments can be supplied in three forms:

1. a length-2 sequence of strings (`file`, `fname`): `file` is the path of a Julia source file that is included into the session, and `fname` the name of a function defined there (at the top level, in `Main`) with the in-place signature `moments!(K, G, theta, U, obj)`. The function must fill `K` (an M -vector holding the counterfactual values $g(U^{(j)}; \theta)$) and `G` (an $M \times d$ matrix holding the moment functions $m(U^{(j)}; \theta)$, one row per draw) and may read the user payload as `obj.gamma` (a Python dict arrives in Julia as a dictionary keyed by symbols, so `obj.gamma[:name]`);
2. a single string naming a Julia function with the same signature that is already defined in the session;
3. a Python function `f(theta, U, gamma)` returning a dict with components `K` (numeric array of length M) and `G` (numeric $M \times d$ array). This path is **much slower** (every objective evaluation crosses the Python/Julia boundary), and because `ForwardDiff` cannot differentiate through Python code the outer optimization needs either a user-supplied gradient or finite differences (the default for this path).

For Julia moments the outer envelope-theorem gradient differentiates the moments by automatic differentiation (`ForwardDiff`), so the Julia function should be written generically in the element type of `theta`; pass `gradient = "fd"` for a non-generic function.

Latent draws. `U` is the $M \times u_dim$ matrix of latent draws $U^{(1)}, \dots, U^{(M)}$ that discretizes the baseline distribution P_* , row j holding the draw $U^{(j)}$. When `U = None` the package generates M scrambled-Halton points in the unit cube $(0, 1)^{d_z}$ of dimension `u_dim` (Owen, 2017), seeded by `seed`; the moments function is then responsible for mapping the uniform coordinates into baseline draws (e.g. through quantile transforms). The Halton generator supports `u_dim` ≤ 15 ; supply `U` directly for higher-dimensional draws.

Divergences. `divergence` selects the entropy function ϕ whose divergence $D_\phi(P \| P_*)$ defines the neighborhood; the budget grid `delta` must be strictly positive, and must lie in $(0, 1]$ for the total-variation family:

- "KL_chi2" (default): the hybrid Kullback-Leibler/chi-square divergence of Christensen and Connault (2023); any `delta` > 0 .
- "KL", "chi2": the pure Kullback-Leibler entropy $\phi_{KL}(s) = s \log s - s + 1$ and the Pearson chi-square entropy; any `delta` > 0 .
- "TV": total variation, $\phi_{TV}(s) = |s - 1|/2$, for which $D_{\phi_{TV}}(P \| P_*) = TV(P, P_*)$. Its recession function is finite, $\phi_{TV}^\infty(1) = 1/2$, so the dual keeps the pointwise constraint $\zeta + \eta/2 \geq \sup_{u \in \mathcal{Z}} \{g(u; \theta) - \lambda^\top m(u; \theta)\}$, which discretizes into M linear feasibility constraints (one per draw).
- "Tvmix": total variation intersected with the mixture (contamination) constraint $P \geq \kappa P_*$, that is, the perturbed distribution contains the baseline as a mixing component with weight $\kappa = 1 - \delta$ (or `control["tvmix_kappa"]`); solved in the exact reduced form (Palomba, 2026).
- "TvmixC": the literal dual of the same program, kept as a cross-check of "Tvmix"; it is slower, and it is the only mode supporting a mixing weight $\kappa < 1 - \delta$.
- "TVac": total variation restricted to distributions absolutely continuous with respect to the baseline, $P \ll P_*$.

The kinked total-variation conjugates ϕ_{TV}^* are Huber-smoothed and the per-draw maxima in the dual objective log-sum-exp-smoothed (scales in `tvbounds_control()`); both smoothings lie above the exact functions, so computed bounds are outward-conservative (wider, never narrower) at order $1e-3$.

Optimization. For each budget (in increasing order) and each side, the solver runs `control["maxsolves"]` multi-start outer optimizations over θ (KNITRO, or Optim.jl when `control["use_optim"] = True`), warm-started at the previous budget's optimum; the reported bound is the inner (dual) value resolved at the best candidate, which makes the bound curves monotone in the budget by construction. A degenerate box (`theta_lb == theta_ub`) skips the outer optimization and reports the bounds at the fixed θ supplied through `theta_init`. Failed budgets are reported as NaN (for "TVmix" a NaN typically signals an infeasible moment condition at every θ in the box, that is an empty robustness set at that budget).

Value

An object of class TVBounds: an object with the fields described in the package overview, in particular bounds (data frame with columns `delta`, `lower`, `upper`, holding δ , $\underline{k}(\delta)$ and $\bar{k}(\delta)$; no standard-error or confidence-band columns, since this application currently carries no inference), `point` (the plug-in counterfactual $\mathbb{E}_{P_*}[g(U; \theta)]$ at `theta_init`, or NaN when `theta_init` was not supplied), `divergence`, and `details`, a dict with:

`solver` data frame of per-budget diagnostics: outer multi-start flags, inner KNITRO status codes, and timings for each side (-999 marks entries that do not apply, e.g. outer flags in fixed-theta mode).

`theta_lower`, `theta_upper` $1 \times \text{length}(\text{delta})$ matrices of the outer-optimal structural parameters θ at each budget, for the lower and the upper bound respectively.

`M`, `u_dim` the number M of simulated draws and their dimension d_z .

`theta_lb`, `theta_ub`, `theta_init`, `fixed_theta` the parameter box, the initial point, and whether the box was degenerate.

`control` the resolved control dict, including the option files actually used.

`moments` a short description of the moments specification.

KNITRO requirement

This function relies on Julia (≥ 1.9) and on the commercial **Artelys KNITRO** solver, accessed through the 'juliacall' package and KNITRO.jl. A valid KNITRO license is required (free academic trials are available from Artelys at <https://www.artelys.com/solvers/knitro/>). On the first call in each Python session the package initializes Julia, instantiates its Julia environment, and checks that KNITRO.jl loads and that a KNITRO solver context can be created (which exercises the license); a one-time message reports the outcome, and the call stops with installation and license guidance when the check fails. The check runs once per Python session.

References

- Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.
- Christensen, T. and B. Connault (2023). "Counterfactual Sensitivity and Robustness." *Econometrica*, 91(1), 263-298.
- Owen, A. B. (2017). "A randomized Halton algorithm in R." arXiv:1706.02808.

See Also

`tvbounds_control()`, `tvbounds_plot()`, `tvbounds_summary()`

Examples

```
# The full solver requires Julia and a licensed KNITRO installation,
# so a complete run cannot be executed without them:
## Not run:
import tvbounds
from tvbounds import tvbounds_counterfactual, tvbounds_control

# Toy model (shipped with the package):  $U \sim \text{Uniform}(0, 1)$ , one moment
# condition  $m(U; \theta) = U - \theta$  and counterfactual  $g(U; \theta) = U$ ,
# so under the "TVmix" neighborhood the bounds equal the endpoints of
# the  $\theta$  box. In the Julia file,  $m$  is written into  $G$  and  $g$  into  $K$ .
toy = tvbounds.julia_file("examples", "toy.jl")
fit = tvbounds_counterfactual(
    moments = (toy, "tvb_toy_moments!"),
    d = 1,
    theta_lb = 0.4, theta_ub = 0.6,
    delta = [0.5, 1],
    divergence = "TVmix",
    M = 500, u_dim = 1,
    theta_init = 0.5,
    control = tvbounds_control(maxsolves = 2),
    seed = 1234)
fit.bounds

## End(Not run)

# The control constructor is pure Python and always available:
from tvbounds import tvbounds_control
tvbounds_control(maxsolves = 3)["maxsolves"]
```

tvbounds_riv

Sensitivity bounds for recentered instrumental variables

Description

Computes sensitivity bounds on a recentered (formula) instrumental-variables estimate when the postulated distribution of the shocks is allowed to vary within a total variation or a contamination neighborhood of the baseline assignment distribution P_* , as in Palomba (2026). The leading use case is the recentered instruments of Borusyak and Hull (2023), whose validity rests on a researcher-postulated distribution for the shock process: the bounds quantify how far the estimate can move when up to a fraction δ of that postulated probability mass is misspecified.

Usage

```
tvbounds_riv(
    y,
    x,
    z,
    Fmat,
    p = None,
    controls = None,
    delta = np.linspace(0, 1, 501),
    neighborhood = "tv",
```

```

    tau_star = 0,
    verbose = False
)

```

Arguments

y	Numeric n-vector, the outcome y_i .
x	Numeric n-vector, the endogenous regressor x_i .
z	Numeric n-vector, the realized (un-recentered) candidate instrument: $z[i]$ is the formula of unit i evaluated at the realized shocks, $z_i = f_i(v; w)$.
Fmat	Numeric $n \times S$ matrix of counterfactual instrument draws: $Fmat[i, s]$ is the formula of unit i evaluated at the s -th counterfactual shock configuration, $f_i(v^{(s)}; w)$. These are the draws from the postulated assignment process used for recentering (in Borusyak and Hull (2023), permuted or re-simulated shock allocations).
p	Optional numeric S -vector holding the probabilities that the postulated assignment distribution P_* attaches to the columns of $Fmat$, that is to the configurations $v^{(1)}, \dots, v^{(S)}$. Defaults to None, meaning the uniform distribution $1/S$ on each draw, which is the postulated assignment distribution of Borusyak and Hull (2023). Must be nonnegative and sum to one.
controls	Optional numeric matrix or data frame of control variables (n rows) to be partialled out of y , x , z , and each column of $Fmat$. A constant is always included. Default None.
delta	Numeric vector of sensitivity budgets $\delta \in [0, 1]$ at which the bounds are traced. Default <code>np.linspace(0, 1, 501)</code> .
neighborhood	Either "tv" (the total variation ball $\mathcal{P}_{TV}^{Fl}(\delta)$, the default) or "contamination" (the contamination neighborhood $\mathcal{P}_{cont}^{Fl}(\delta)$).
tau_star	Numeric scalar reference value τ_* for the breakdown budget $\delta_b(\tau_*)$: the smallest budget at which the bounds cover τ_* . The default 0 gives the breakdown budget for the sign of β .
verbose	Logical; if True, print progress messages. Default False.

Details

The exercise is conducted conditionally on the realized sample, so every bound is a deterministic function of the data and of the budget δ ; accordingly, and by design (as in the paper), **no standard errors or confidence bands are produced** for this application.

The model and the recentered estimate. For units $i \in [n]$ the structural equation is

$$y_i = \beta x_i + \varepsilon_i,$$

with β the parameter of interest, y_i the outcome, x_i the endogenous regressor and ε_i the unobserved residual. Let v denote the vector of exogenous shocks, taking values in a space $\mathcal{V}$, let w collect predetermined covariates, and let $f_i(\cdot; w) : \mathcal{V} \rightarrow \mathbb{R}$ be the known formula that maps a shock configuration into the instrument of unit i , so that $z_i := f_i(v; w)$ is the candidate instrument at the realized shocks. For a distribution P on $\mathcal{V}$, the expected instrument and the recentered instrument are

$$\mu_i(P) := \mathbb{E}_P[f_i(v; w) \mid w] = \int_{\mathcal{V}} f_i(v'; w) dP(v'), \quad \tilde{z}_i(P) := z_i - \mu_i(P).$$

Borusyak and Hull (2023) postulate an assignment distribution P_* for the shocks and recenter at it.

The two criterion functions. The formula enters only through the two sample aggregates

$$g_y(\cdot) := \sum_{i=1}^n y_i f_i(\cdot; w), \quad g_x(\cdot) := \sum_{i=1}^n x_i f_i(\cdot; w),$$

whose recentered values are the reduced form $G_y(P) := g_y(v) - \mathbb{E}_P[g_y]$ and the first stage $G_x(P) := g_x(v) - \mathbb{E}_P[g_x]$. The estimate at a candidate assignment distribution is the ratio

$$\hat{\beta}(P) := \frac{G_y(P)}{G_x(P)} = \frac{\sum_{i=1}^n \tilde{z}_i(P) y_i}{\sum_{i=1}^n \tilde{z}_i(P) x_i},$$

and the reported estimate is $\hat{\beta}_* = \hat{\beta}(P_*)$.

How the arguments encode the shock space. The package represents P_* by S shock configurations $v^{(1)}, \dots, v^{(S)}$: entry $\text{Fmat}[i, s]$ holds $f_i(v^{(s)}; w)$, the formula of unit i at the s -th configuration, $\text{p}[s]$ holds the probability that P_* assigns to that configuration, and $\text{z}[i]$ holds $z_i = f_i(v; w)$. The shock space is therefore taken to be the finite set $\mathcal{V} = \{v^{(1)}, \dots, v^{(S)}\}$, over which the candidate distributions P range.

Neighborhoods. The bounds report the range of $\hat{\beta}(P)$ as P ranges over the chosen neighborhood of P_* , intersected with the set $\mathcal{P}_{\neq 0} := \{P \in \Delta(\mathcal{V}) : G_x(P) \neq 0\}$ of distributions at which the ratio is defined:

- neighborhood = "tv" (the default): the total variation ball

$$\mathcal{P}_{\text{TV}}^{\text{FI}}(\delta) := \{P \in \Delta(\mathcal{V}) : \text{TV}(P, P_*) \leq \delta\},$$

which delivers the bounds $\underline{\beta}_{\text{TV}}(\delta)$ and $\overline{\beta}_{\text{TV}}(\delta)$. They are computed from the closed form the paper gives for the criterion bounds $\underline{g}_{\text{TV}}(h; \delta)$ and $\overline{g}_{\text{TV}}(h; \delta)$, the smallest and the largest value of $\mathbb{E}_P[h]$ over the ball: writing q_h for the quantile function of a criterion h under P_* , and $\overline{h}, \underline{h}$ for its extremes over $\mathcal{V}$,

$$\overline{g}_{\text{TV}}(h; \delta) = \delta \overline{h} + \int_{\delta}^1 q_h(u) du, \quad \underline{g}_{\text{TV}}(h; \delta) = \delta \underline{h} + \int_0^{1-\delta} q_h(u) du,$$

that is, the baseline mean of h once a tail of mass δ has been trimmed and relocated to the most (least) favorable configuration. Applied to the one-parameter family of criteria $g_b := g_y - b g_x$, for which $\hat{\beta}(P) = b$ holds exactly when $\mathbb{E}_P[g_b] = g_b(v)$, each bound is the unique root in b of a strictly monotone function and is located by bracketed root finding;

- neighborhood = "contamination": the contamination neighborhood

$$\mathcal{P}_{\text{cont}}^{\text{FI}}(\delta) := \{P \in \Delta(\mathcal{V}) : P = \delta R + (1 - \delta)P_*, R \in \Delta(\mathcal{V})\},$$

in which the contamination share is the budget δ itself, delivering the bounds $\underline{\beta}_{\text{cont}}(\delta)$ and $\overline{\beta}_{\text{cont}}(\delta)$. These are attained by contaminating distributions R degenerate at a single shock configuration, so they are obtained by enumerating the S configurations rather than by optimization.

Every member of $\mathcal{P}_{\text{cont}}^{\text{FI}}(\delta)$ lies in $\mathcal{P}_{\text{TV}}^{\text{FI}}(\delta)$, so the contamination bounds are weakly tighter at every budget.

First-stage breakdown. Once the budget is large enough that some distribution in the neighborhood makes the recentered first stage $G_x(P)$ vanish, $\hat{\beta}(P)$ is no longer well defined over the whole neighborhood and the identified set is the entire real line. The smallest such budget is the first-stage breakdown budget, $\delta_{\text{FS}}^{\text{TV}}$ for the total variation ball and $\delta_{\text{FS}}^{\text{cont}}$ for the contamination neighborhood; since the contamination neighborhood is the smaller of the two, $\delta_{\text{FS}}^{\text{TV}} \leq \delta_{\text{FS}}^{\text{cont}}$. The one for the

neighborhood in use is reported in `details["delta_fs"]`. Following the paper, the infimum over an empty set is set to 1, so a reported value of 1 carrying `delta_fs_censored = True` means that the first stage never breaks down over the budget range, not that breakdown occurs at 1. Rows of bounds at budgets where the bounds are vacuous carry NaN.

Controls. When `controls` is supplied, y , x , z , and every column of F_{mat} are residualized on the controls and a constant (when `controls = None`, on the constant alone) before the bounds are computed. By the Frisch–Waugh–Lovell theorem this leaves the just-identified two-stage least squares coefficient on x unchanged, because the projection matrix is idempotent; the baseline estimate then replicates the estimate from the full regression with controls.

Value

An object of class `TVBounds`, an object with attributes

- `application`: "riv".
- `bounds`: data frame with one row per budget and columns `delta`, `lower`, `upper`. Rows at budgets where the neighborhood contains a distribution collapsing the recentered first stage carry NaN (the bounds are vacuous there). No standard error or confidence interval columns are attached: this application carries no inference by design.
- `point`: the reported recentered IV estimate $\hat{\beta}_* = \hat{\beta}(P_*)$, which is the common value of the two bounds at $\text{delta} = 0$.
- `n`: number of observations.
- `neighborhood`: the neighborhood used.
- `level`, `B`: NaN (no inference).
- `estimand_label`: "IV coefficient".
- `call`: the matched call.
- `details`: a dict with
 - `criteria`: the reduced criterion object consumed by the internal kernels, holding the realized values $g_y(v)$ and $g_x(v)$, the two S-vectors $g_y(v^{(s)})$ and $g_x(v^{(s)})$, and the baseline aggregates $\mathbb{E}_{P_*}[g_y]$ and $\mathbb{E}_{P_*}[g_x]$; bounds at additional budgets can be recomputed from it without touching the $n \times S$ design again.
 - `delta_fs`, `delta_fs_censored`: the first-stage breakdown budget for the neighborhood used, $\delta_{\text{FS}}^{\text{TV}}$ or $\delta_{\text{FS}}^{\text{cont}}$, and whether it is censored at 1 (True when the first stage never breaks down on $[0, 1]$, so 1 is the empty-set convention rather than an actual breakdown).
 - `delta_fs_tv`, `delta_fs_cont`: both first-stage breakdown budgets, $\delta_{\text{FS}}^{\text{TV}}$ and $\delta_{\text{FS}}^{\text{cont}}$ (each carrying its censored attribute).
 - `delta_breakdown`, `delta_breakdown_censored`: the smallest budget $\delta_b(\tau_*)$ at which the bounds for the chosen neighborhood cover τ_* , reported as 1 with `delta_breakdown_censored = True` when τ_* is never covered on $[0, 1]$.
 - `tau_star`: the reference value τ_* used.
 - `n_controls`: number of control columns partialled out (0 when `controls = None`).

References

- Palomba, F. (2026). "Sensitivity Analysis in Population Shares." Working paper.
- Borusyak, K. and Hull, P. (2023). "Nonrandom Exposure to Exogenous Shocks." *Econometrica*, 91(6), 2155–2185.

Examples

```

import numpy as np
import pandas as pd
from tvbounds import tvbounds_riv

# A small simulated formula-instrument design: S counterfactual shock
# configurations, a realized instrument, a first stage, and an outcome.
rng = np.random.default_rng(123)
n = 80
S = 50
Fmat = rng.normal(size = (n, S))          # Fmat[i, s] = f_i(v^(s); w)
z = Fmat.mean(axis = 1) + rng.normal(size = n) # realized instrument z_i = f_i(v; w)
x = z + 0.5 * rng.normal(size = n)         # endogenous regressor
y = 0.4 * x + rng.normal(size = n)         # outcome

fit = tvbounds_riv(y, x, z, Fmat, delta = np.linspace(0, 1, 21))
fit.point                                # recentered IV estimate at P_*
fit.bounds.head()                       # bounds along the budget grid
fit.details["delta_fs"]                 # first-stage breakdown budget
fit.details["delta_breakdown"]          # breakdown budget for the sign

# Contamination neighborhood, with controls partialled out.
W = pd.DataFrame({"w1": rng.normal(size = n), "w2": rng.normal(size = n)})
fit_cont = tvbounds_riv(y, x, z, Fmat, controls = W,
                        delta = np.linspace(0, 1, 21),
                        neighborhood = "contamination")
fit_cont.point

```

Index

`plot.tvbounds`, [2](#)
`print.tvbounds`, [4](#)
`print.tvbounds_summary`, [5](#)

`summary.tvbounds`, [6](#)

`tvbounds_attrition`, [9](#)
`tvbounds_attrition()`, [2](#), [6](#)
`tvbounds_control`, [14](#)
`tvbounds_control()`, [18–20](#)
`tvbounds_counterfactual`, [17](#)
`tvbounds_counterfactual()`, [2](#), [6](#), [14](#), [16](#)
`tvbounds_plot` (`plot.tvbounds`), [2](#)
`tvbounds_plot()`, [9](#), [13](#), [20](#)
`tvbounds_riv`, [21](#)
`tvbounds_riv()`, [2](#), [6](#)
`tvbounds_summary` (`summary.tvbounds`), [6](#)
`tvbounds_summary()`, [3](#), [5](#), [13](#), [20](#)