Package: confoundvis (via r-universe)

June 1, 2026

Type Package

Title Visualization Tools for Sensitivity Analysis of Unmeasured Confounding

Version 0.1.0

Description Provides visualization tools for sensitivity analysis to unmeasured confounding in observational studies. Includes contour-based sensitivity plots, robustness curves, and benchmark-oriented graphics that help researchers assess how strong omitted confounding would need to be to attenuate, invalidate, or reverse estimated effects. Supports regression-based sensitivity analysis frameworks, including impact threshold approaches (Frank, 2000, <doi:10.1177/0049124100029002001>), partial R-squared methods (Cinelli and Hazlett, 2020, <doi:10.1111/rssb.12348>), and E-value style metrics (VanderWeele and Ding, 2017, <doi:10.7326/M16-2607>). Emphasizes clear, interpretable, and publication-ready graphical summaries for transparent reporting of causal sensitivity analyses across the social, behavioral, health, and educational sciences.

License GPL-3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.4.0), rlang, graphics, stats

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

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/causalfragility-lab/confoundvis

BugReports https://github.com/causalfragility-lab/confoundvis/issues

Repository https://causalfragility-lab.r-universe.dev

Date/Publication 2026-03-30 16:52:37 UTC

RemoteUrl https://github.com/causalfragility-lab/confoundvis

RemoteRef HEAD

RemoteSha bd9347316b24d0b801d229474918f7a2e471758f

Contents

`as.data.frame.confoundsens` *Coerce a confoundsens object to a data frame*

Description

Coerce a confoundsens object to a data frame

Usage

```
## S3 method for class 'confoundsens'
as.data.frame(x, ...)
```

Arguments

`x`	A `confoundsens` object.
`...`	Unused; included for S3 method consistency.

Value

A data.frame with columns `lambda` and `theta`, plus optional columns `level`, `se`, and `t` when present in `x`.

Examples

```
x <- new_confoundsens(
  lambda = seq(0, 0.2, length.out = 5),
  theta  = seq(1, 0.8, length.out = 5),
  se     = rep(0.05, 5)
)
as.data.frame(x)
```

`as_confoundsens` *Convert a data frame to a confoundsens object*

Description

Converts a data frame containing sensitivity analysis results into a `confoundsens` object suitable for use with `confoundvis` plotting functions.

Usage

```
as_confoundsens(
  data,
  lambda = "lambda",
  theta = "theta",
  level = NULL,
  se = NULL,
  t = NULL
)
```

Arguments

`data`	A data.frame containing sensitivity analysis results.
`lambda`	Character string; name of the column containing lambda values. Default `"lambda"`.
`theta`	Character string; name of the column containing theta values. Default `"theta"`.
`level`	Optional character string; name of the column containing level identifiers (e.g., `"within"` / `"between"`).
`se`	Optional character string; name of the column containing standard errors for `theta`.
`t`	Optional character string; name of the column containing test statistics.

Value

A `confoundsens` object.

Examples

```
df <- data.frame(
  lambda = seq(0, 0.2, length.out = 10),
  theta  = seq(1, 0.5, length.out = 10),
  se     = rep(0.1, 10),
  level  = rep(c("within", "between"), length.out = 10)
)
x <- as_confoundsens(df)
x
```

`fit_local_quadratic` *Fit a local quadratic approximation*

Description

Fits a second-order Taylor (quadratic) approximation of an effect path $\theta(\delta)$ near $\delta = 0$ using ordinary least squares:

$$\theta(\delta) \approx a + b\delta + c\delta^2.$$

Usage

```
fit_local_quadratic(
  data = NULL,
  delta = NULL,
  theta = NULL,
  local_max_delta = 0.2,
  include_intercept = TRUE,
  tol = 1e-12
)
```

Arguments

`data`	Optional data.frame containing columns named `delta` and `theta`. If supplied, the `delta` and `theta` arguments are ignored.
`delta`	Optional numeric vector of delta values. Used only when `data = NULL`.
`theta`	Optional numeric vector of theta values. Used only when `data = NULL`.
`local_max_delta`	Positive numeric scalar giving the half-width of the local window: only observations with $\lvert\delta\rvert \leq$ `local_max_delta` are used in the fit.
`include_intercept`	Logical; if `TRUE` (default), include an intercept term.
`tol`	Non-negative numeric scalar tolerance used when selecting observations within the local window.

Details

You may supply either a data.frame containing columns `delta` and `theta`, or supply numeric vectors `delta` and `theta` directly.

Value

A named list with elements:

- `coef` — named coefficient vector from `stats::lm()`.
- `intercept`, `slope`, `quad` — individual coefficients (NA when absent).
- `model` — the fitted `lm` object.
- `local_data` — the data.frame used for fitting.
- `local_max_delta` — the window half-width used.

Examples

```
df <- data.frame(
  delta = seq(0, 0.3, length.out = 60),
  theta = 0.4 - 0.7 * seq(0, 0.3, length.out = 60) +
          0.4 * seq(0, 0.3, length.out = 60)^2
)
fit_local_quadratic(df, local_max_delta = 0.2)
```

---

new_confoundsens *Create a confoundsens object*

Description

Constructs a `confoundsens` object — a lightweight container for storing sensitivity paths (e.g., $\theta(\lambda)$) and optional uncertainty or stratification information used by `confoundvis` plotting functions.

Usage

```
new_confoundsens(lambda, theta, level = NULL, se = NULL, t = NULL)
```

Arguments

`lambda`	Numeric vector of sensitivity strength values (e.g., ITCV lambda or delta). Should be nondecreasing; a warning is issued otherwise.
`theta`	Numeric vector of effect estimates along the sensitivity path. Must be the same length as `lambda`.
`level`	Optional character (or coercible) vector of level identifiers (e.g., `"within"`, `"between"`). Must be the same length as `lambda`.
`se`	Optional numeric vector of standard errors for `theta`. Must be the same length as `lambda`.
`t`	Optional numeric vector of test statistics along the path. Must be the same length as `lambda`.

Value

A confoundsens object (a list with class "confoundsens").

Examples

```
x <- new_confoundsens(
  lambda = seq(0, 0.2, length.out = 10),
  theta  = seq(1, 0.6, length.out = 10),
  se     = rep(0.1, 10)
)
x
```

---

plot_cone_comparison *Compare reversal cones across multilevel components*

Description

Produces a two-panel plot comparing the reversal cone cross-sections for the within-cluster and between-cluster confounding components in multilevel settings. Each panel calls plot_reversal_cone() and they are displayed side by side using faceting.

Usage

```
plot_cone_comparison(
  theta0_within = 0.35,
  theta0_between = 0.5,
  delta_within = 0.2,
  delta_between = 0.3,
  ...
)
```

Arguments

theta0_within	Numeric; within-cluster baseline effect.
theta0_between	Numeric; between-cluster baseline effect.
delta_within	Numeric; within-cluster confounding magnitude (> 0).
delta_between	Numeric; between-cluster confounding magnitude (> 0).
...	Additional arguments passed to plot_reversal_cone().

Value

Invisibly, a list with ggplot objects within and between. The plots are drawn side-by-side to the current device.

Examples

```
plot_cone_comparison(
  theta0_within  = 0.35, theta0_between = 0.50,
  delta_within   = 0.20, delta_between  = 0.30
)
```

`plot_figure2_taylor_panels` *Three-panel Taylor approximation figure*

Description

Produces a three-panel figure (linear / concave-down / convex-up) showing a simulated confounding path together with its first-order (tangent) and second-order (local quadratic) approximations. The panels correspond to the three curvature regimes in the mITCV framework.

Usage

```
plot_figure2_taylor_panels(
  delta_max = 1.5,
  step = 0.02,
  theta0 = 0.4,
  slope = -0.7,
  kappa = 0.4,
  local_max_delta = 0.2
)
```

Arguments

`delta_max`	Positive numeric scalar. Upper bound of the delta grid.
`step`	Positive numeric scalar. Grid step size.
`theta0`	Numeric scalar. Baseline effect at delta = 0.
`slope`	Numeric scalar. First-order slope at delta = 0.
`kappa`	Non-negative numeric scalar. Curvature magnitude.
`local_max_delta`	Positive numeric scalar <= `delta_max`. Width of the local window used for the quadratic approximation.

Value

A named list with ggplot objects `A` (linear), `B` (concave), and `C` (convex), returned invisibly. The three panels are also printed to the active graphics device via `gridExtra::grid.arrange()` if **gridExtra** is installed, or via `graphics::layout()` otherwise.

Examples

```
plots <- plot_figure2_taylor_panels()
# Access individual panels
plots$A
```

`plot_local_taylor` *Local Taylor diagnostic plot*

Description

Plots local Taylor series components (or any multi-series decomposition) as a function of `delta` from a long-form data.frame. Lines are distinguished by colour and linetype, keyed by `series`.

Usage

```
plot_local_taylor(df, facet = FALSE)
```

Arguments

`df`	A data.frame in long form with columns: • `delta` — numeric; the confounding-strength grid. • `series` — character or factor; name of each curve (e.g., `"path"`, `"tangent"`, `"quadratic"`). • `value` — numeric; the effect value for each (delta, series) pair.
`facet`	Logical; if `TRUE`, produce a faceted plot with one panel per `series`. If `FALSE` (default), overlay all series on a single panel with colour and linetype aesthetics.

Value

A `ggplot2::ggplot()` object.

Examples

```
df <- data.frame(
  delta  = rep(seq(0, 0.2, length.out = 25), 3),
  series = rep(c("path", "tangent", "quadratic"), each = 25),
  value  = c(
    0.4 - 0.7 * seq(0, 0.2, length.out = 25) - 0.4 * seq(0, 0.2, length.out = 25)^2,
    0.4 - 0.7 * seq(0, 0.2, length.out = 25),
    0.4 - 0.7 * seq(0, 0.2, length.out = 25) + 0.2 * seq(0, 0.2, length.out = 25)^2
  )
)
plot_local_taylor(df)
plot_local_taylor(df, facet = TRUE)
```

`plot_reversal_cone` *Reversal cone geometry plot*

Description

Visualizes the two-dimensional cross-section of the reversal cone C_l at a fixed confounding effect magnitude $|\delta|$. The cone partitions the (q, p) confounding parameter space into an **attenuation zone** (where the effect shrinks but does not reverse sign) and a **reversal zone** (where the sign changes). The boundary between zones is the reversal curve derived from the multilevel mITCV framework.

Usage

```
plot_reversal_cone(
  theta0 = 0.4,
  delta = 0.2,
  q_range = c(0, 1),
  p_range = c(-1, 1),
  grid_n = 200L,
  show_boundary = TRUE,
  show_volume = TRUE
)
```

Arguments

`theta0`	Numeric; baseline estimated effect at zero confounding (must be nonzero).
`delta`	Numeric; the fixed confounding effect magnitude $\|\delta\|$ at which the cross-section is evaluated (> 0).
`q_range`	Numeric vector of length 2; range of the confounding prevalence parameter $q \in [0, 1]$ (default `c(0, 1)`).
`p_range`	Numeric vector of length 2; range of the confounding impact parameter p (default `c(-1, 1)`).
`grid_n`	Integer; grid resolution for each axis (default 200).
`show_boundary`	Logical; draw the reversal boundary curve.
`show_volume`	Logical; annotate with the cone volume proportion.

Value

A ggplot object.

Examples

```
plot_reversal_cone(theta0 = 0.40, delta = 0.20)
plot_reversal_cone(theta0 = 0.40, delta = 0.50)
```

`plot_robustness_curve` *Plot a robustness curve*

Description

Plots the sensitivity path stored in a `confoundsens` object as a function of `lambda`. By default, plots the effect path `theta(lambda)`; optionally plots the test-statistic path `t(lambda)`. Pointwise 95% confidence bands are drawn when standard errors are available.

Usage

```
plot_robustness_curve(
  x,
  what = c("theta", "t"),
  bands = TRUE,
  points = TRUE,
  facet_level = TRUE
)
```

Arguments

`x`	A `confoundsens` object created by `new_confoundsens()` or `as_confoundsens()`.
`what`	Character; which path to plot: `"theta"` (default) or `"t"`.
`bands`	Logical; if `TRUE` and `x$se` is present, draw 95% pointwise confidence bands (applies only when `what = "theta"`).
`points`	Logical; if `TRUE`, overlay points on the line.
`facet_level`	Logical; if `TRUE` and `x$level` is present, facet the plot by level.

Value

A `ggplot2::ggplot()` object.

Examples

```
x <- new_confoundsens(
  lambda = seq(0, 0.2, length.out = 25),
  theta  = 1 - 2 * seq(0, 0.2, length.out = 25),
  se     = rep(0.05, 25),
  level  = rep(c("within", "between"), length.out = 25)
)
plot_robustness_curve(x)
plot_robustness_curve(x, bands = FALSE, facet_level = FALSE)
```

plot_sensitivity_contour *Sensitivity contour plot*

Description

Draws an ITCV-style hyperbolic boundary in (r_{YU}, r_{DU}) space and optionally overlays observed covariate benchmarks as labelled points. The robust region is the interior of the hyperbola where $|r_{YU} \cdot r_{DU}| <$ `threshold`.

Usage

```
plot_sensitivity_contour(threshold, grid_n = 200, benchmarks = NULL)
```

Arguments

`threshold`	Positive numeric scalar; the ITCV-style product threshold. The boundary satisfies $\|r_{YU} \cdot r_{DU}\| =$ `threshold`.
`grid_n`	Integer >= 50; number of points used per branch of the boundary curve. Larger values give smoother curves.
`benchmarks`	Optional data.frame with columns `r_yu` and `r_du` (numeric) and an optional `label` column (character). Each row is plotted as a labelled benchmark point.

Value

A `ggplot2::ggplot()` object.

Examples

```
b <- data.frame(
  r_yu  = c(0.10, 0.15),
  r_du  = c(0.20, 0.12),
  label = c("SES", "BMI")
)
plot_sensitivity_contour(threshold = 0.02, benchmarks = b)
```

plot_sensitivity_love *Sensitivity Love plot*

Description

Benchmarks a sensitivity threshold against the empirical distribution of observed covariate impacts in a "Love plot"-style display. Each covariate appears as a point on a horizontal impact axis; the sensitivity threshold is shown as a vertical reference line. Covariates to the left of the line are weaker than the threshold; those to the right pose a credible threat.

Usage

```
plot_sensitivity_love(df, threshold, sort = TRUE, top = NULL)
```

Arguments

df	A data.frame with at least two columns: • covariate — covariate names (character or factor). • impact — numeric impact values (e.g., ITCV product $\lvert r_{YU} \cdot r_{DU} \rvert$, partial R^2, or other confounding-strength metric).
threshold	Single non-missing numeric value to draw as a vertical reference line (the sensitivity threshold).
sort	Logical; if TRUE (default), sort covariates by impact on the y-axis (ascending).
top	Optional positive integer. If supplied, only the top covariates with the largest absolute impact are displayed.

Value

A ggplot2::ggplot() object.

Examples

```
df <- data.frame(
  covariate = c("SES", "BMI", "Gender", "Race", "Age"),
  impact    = c(0.12, 0.05, 0.02, 0.08, 0.03)
)
plot_sensitivity_love(df, threshold = 0.10)
plot_sensitivity_love(df, threshold = 0.10, top = 3)
```

---

print.confoundsens *Print method for confoundsens objects*

Description

Prints a concise summary of the key fields in a confoundsens object.

Usage

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

Arguments

x	A confoundsens object.
...	Unused; included for S3 method consistency.

Value

x, invisibly.

Examples

```
x <- new_confoundsens(
  lambda = seq(0, 0.2, length.out = 5),
  theta  = seq(1, 0.8, length.out = 5)
)
print(x)
```

`simulate_taylor_demo` *Simulate demo confounding paths for Taylor panel figures*

Description

Generates three synthetic sensitivity paths — linear, concave-down, and convex-up — sharing the same baseline effect $\theta(0)$ and first-order slope, but differing in second-order curvature. These toy paths illustrate the difference between tangent-based (first-order) and local quadratic (second-order) sensitivity approximations.

Usage

```
simulate_taylor_demo(
  delta_max = 1.5,
  step = 0.02,
  theta0 = 0.4,
  slope = -0.7,
  kappa = 0.4
)
```

Arguments

`delta_max`	Single positive numeric scalar. Upper bound of the δ grid.
`step`	Single positive numeric scalar. Step size for the grid.
`theta0`	Single numeric scalar. Baseline effect at $\delta = 0$.
`slope`	Single numeric scalar. First-order slope at $\delta = 0$.
`kappa`	Single non-negative numeric scalar. Curvature magnitude.

Details

The three regimes are:

- **Linear**: $\theta(\delta) = \theta_0 + s\delta$
- **Concave-down**: $\theta(\delta) = \theta_0 + s\delta - \kappa\delta^2$
- **Convex-up**: $\theta(\delta) = \theta_0 + s\delta + \kappa\delta^2$

Value

A named list with elements `linear`, `concave`, and `convex`, each a `new_confoundsens()` object.

Examples

```
sims <- simulate_taylor_demo(
  delta_max = 1, step = 0.05, theta0 = 0.4, slope = -0.7, kappa = 0.4
)
sims$linear
plot_robustness_curve(sims$concave)
```

---

summary.confoundsens *Summary method for confoundsens objects*

Description

Produces a concise numerical summary of the sensitivity path, optionally broken down by level.

Usage

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

Arguments

`object`	A `confoundsens` object.
`...`	Unused; included for S3 method consistency.

Value

An object of class `summary.confoundsens` containing a `table` data.frame with per-level summary statistics.

Examples

```
x <- new_confoundsens(
  lambda = seq(0, 0.2, length.out = 10),
  theta  = seq(1, 0.6, length.out = 10),
  se     = rep(0.08, 10),
  level  = rep(c("within", "between"), length.out = 10)
)
summary(x)
```

Index