Package: causalfrag (via r-universe)

June 16, 2026

Type Package

Title Cross-Framework Causal Fragility Index

Version 0.1.1

Date 2026-05-30

Description Provides a unified workflow for running, classifying, visualizing, and interpreting sensitivity analyses for unmeasured confounding across multiple causal frameworks. Introduces the Causal Fragility Index (CFI), a single 0-100 composite score that integrates evidence from the partial R-squared robustness value approach (Cinelli and Hazlett, 2020, <doi:10.1111/rssb.12348>), E-value metrics (VanderWeele and Ding, 2017, <doi:10.7326/M16-2607>), and the Impact Threshold for a Confounding Variable (Frank, 2000, <doi:10.1177/0049124100029002001>) into one interpretable measure of robustness. The package also provides template-based plain-language narrative interpretation and publication-ready reporting, with optional integration with the 'confoundvis' package for sensitivity plots.

License MIT + file LICENSE

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

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

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports cli (>= 3.4.0), rlang (>= 1.0.0), jsonlite (>= 1.8.0), glue (>= 1.6.0)

Suggests httr2 (>= 1.0.0), confoundvis (>= 0.1.0), sensemakr (>= 0.1.4), EValue (>= 4.1.3), konfound (>= 0.4.0), rbounds (>= 2.1), rmarkdown (>= 2.14), knitr (>= 1.39), testthat (>= 3.0.0), withr (>= 2.5.0)

VignetteBuilder knitr

Config/testthat/edition 3

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

Date/Publication 2026-06-06 23:30:35 UTC

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

RemoteRef HEAD

RemoteSha 9e9393f04d804510e61f0cfafc695d77bbbba833

Contents

`causalfrag-package` *causalfrag: A Cross-Framework Causal Fragility Index for Sensitivity Analysis*

Description

causalfrag provides a unified workflow for running, classifying, visualizing, and interpreting sensitivity analyses for unmeasured confounding across multiple causal frameworks.

The core contribution is the **Causal Fragility Index (CFI)**: a single 0–100 composite score that integrates sensitivity evidence across frameworks into one interpretable measure of robustness.

Main workflow

“‘r fit <- lm(outcome ~ treatment + covariates, data = mydata) res <- run_sensitivity(fit, treatment = "treatment", data = mydata) res <- flag_fragility(res) # also computes CFI automatically print(res) # shows CFI score + component breakdown print_cfi(res) # dedicated CFI display generate_report(res) # full structured report “‘

Causal Fragility Index (CFI)

The CFI combines evidence from up to four sensitivity components:

- **RV** (point): Robustness Value for point-estimate nullification
- **RV** (significance): Robustness Value for statistical significance
- **E-value**: Risk-ratio style confounding strength
- **Pct bias**: Percent bias needed to invalidate (ITCV)

Supported sensitivity frameworks

- **sensemakr**: Cinelli & Hazlett (2020) partial R-squared approach
- **EValue**: VanderWeele & Ding (2017) E-value metrics
- **konfound**: Frank (2000) Impact Threshold (ITCV)
- **rbounds**: Rosenbaum (2002) bounds for matched designs

Relationship to confoundvis

confoundvis (Hait, 2026) provides visualization tools for sensitivity analysis. **causalfrag** provides the unified fragility scoring, interpretation, and reporting layer that optionally uses **confoundvis** graphics as part of the workflow.

Author(s)

Subir Hait <haitsubi@msu.edu>

References

Frank, K. A. (2000). Impact of a confounding variable on the inference of a regression coefficient. *Sociological Methods & Research*, 29(2), 147–194. doi:10.1177/0049124100029002001

Cinelli, C., & Hazlett, C. (2020). Making sense of sensitivity: Extending omitted variable bias. *Journal of the Royal Statistical Society: Series B*, 82(1), 39–67. doi:10.1111/rssb.12348

VanderWeele, T. J., & Ding, P. (2017). Sensitivity analysis in observational research: Introducing the E-value. *Annals of Internal Medicine*, 167(4), 268–274. doi:10.7326/M162607

See Also

Useful links:

- https://github.com/causalfragility-lab/causalfrag
- Report bugs at https://github.com/causalfragility-lab/causalfrag/issues

`compare_cfi` *Compare CFI across multiple treatment effects*

Description

Produces a side-by-side comparison table of CFI scores and fragility classifications across two or more `sens_results` objects. Useful for comparing policy interventions or treatment arms.

Usage

```
compare_cfi(...)
```

Arguments

`...`	Two or more named `sens_results` objects. Names become the row labels in the output table.

Value

A data frame with one row per treatment and columns: `treatment`, `outcome`, `cfi`, `label`, `point_fragility`, `significance_fragility`, `rv_q`, `evalue`, `pct_bias`.

Examples

```
if (requireNamespace("sensemakr", quietly = TRUE)) {
  data("darfur", package = "sensemakr")

  fit_a <- lm(peacefactor ~ directlyharmed + age + female, data = darfur)
  fit_b <- lm(peacefactor ~ herder_dar    + age + female, data = darfur)

  res_a <- run_sensitivity(fit_a, "directlyharmed", darfur, verbose = FALSE)
  res_b <- run_sensitivity(fit_b, "herder_dar",     darfur, verbose = FALSE)

  res_a <- flag_fragility(res_a)
  res_b <- flag_fragility(res_b)

  compare_cfi(
    "Directly harmed" = res_a,
    "Herder"          = res_b
  )
}
```

compute_cfi *Compute the Causal Fragility Index (CFI)*

Description

The Causal Fragility Index (CFI) is a single 0–100 composite score that integrates sensitivity evidence across multiple causal frameworks into one interpretable measure of robustness.

Each framework contributes one or more component scores, which are averaged into the final CFI. Higher scores indicate greater robustness to unmeasured confounding.

Usage

```
compute_cfi(x, weights = NULL)
```

Arguments

x	A sens_results object from run_sensitivity().
weights	Optional named numeric vector to weight components. Names must match: "rv", "rv_alpha", "evalue", "pct_bias". Default: equal weights.

Value

The input sens_results object with $cfi populated as a named list with elements:

score Numeric 0–100. The overall CFI.

label Character. CFI classification label.

components Named numeric vector of component scores.

interpretation Character. One-sentence plain-language summary.

Scoring rules

Component	Rule	Full score at
RV (point)	rv / 0.20 * 100	RV >= 0.20
RV (significance)	rv_alpha / 0.10 * 100	RV >= 0.10
E-value	(evalue - 1) / 2.0 * 100	E-value >= 3
Percent bias	pct_bias (direct)	pct >= 100

CFI classification

CFI range	Label
0 – 24	Fragile
25 – 49	Moderately fragile
50 – 74	Moderately robust
75 – 100	Robust

Examples

```
res <- new_sens_results(
  design = "regression", treatment = "directlyharmed",
  outcome = "peacefactor", frameworks = c("sensemakr", "evalue", "itcv"),
  results = list(
    sensemakr = list(rv_q = 0.139, rv_qa = 0.076,
                     r2yd_x = 0.022, estimate = 0.097),
    evalue    = list(evalue = 1.90, evalue_lower = 1.55,
                     estimate = 0.097),
    itcv      = list(itcv = 0.065, rir = 678, pct_bias = 53.1)
  )
)
res <- compute_cfi(res)
print_cfi(res)
```

---

compute_cfi_benchmarked

Compute benchmark-adjusted CFI

Description

Extends the CFI with benchmark-relative interpretation. Rather than assessing fragility against absolute thresholds, the benchmark-adjusted CFI compares the required confounding strength to the observed association of a named covariate with treatment and outcome.

This answers: "Would an omitted confounder need to be stronger than `benchmark_covariate` to overturn the conclusion?"

Usage

```
compute_cfi_benchmarked(x, benchmark_covariate, kd = 1)
```

Arguments

x	A sens_results object with sensemakr results.
benchmark_covariate	Character. Name of the covariate to benchmark against. Must be a covariate in the original model.
kd	Numeric. Multiplier for treatment partial R-squared in benchmark. Default 1 (benchmark at 1x covariate strength).

Value

The input sens_results object with $cfi_benchmarked populated as a named list with elements:

rv_q_benchmark Ratio: RV / benchmark partial R-squared (treat).

rv_qa_benchmark Same for significance threshold.

benchmark_label Plain-language benchmark statement.

exceeds_benchmark Logical. Does required confounding exceed the benchmark covariate's strength?

Examples

```
if (requireNamespace("sensemakr", quietly = TRUE)) {
  data("darfur", package = "sensemakr")
  fit <- lm(peacefactor ~ directlyharmed + age + female + village,
            data = darfur)
  res <- run_sensitivity(fit, treatment = "directlyharmed", data = darfur,
                         benchmark_covariates = "female")
  res <- compute_cfi_benchmarked(res, benchmark_covariate = "female")
  cat(res$cfi_benchmarked$benchmark_label)
}
```

---

compute_cfi_weighted *Compute CFI with user-defined or preset weights*

Description

Extends compute_cfi() with named weight presets and user-defined weights. Different sensitivity frameworks carry different evidential weight depending on the study design and analyst priorities.

Usage

```
compute_cfi_weighted(x, weights = "balanced")
```

Arguments

x	A sens_results object from run_sensitivity().
weights	Character preset or named numeric vector. Character: one of "balanced", "rv_priority", "evalue_priority", "itcv_priority". Numeric: named vector with any subset of c("rv", "rv_alpha", "evalue", "pct_bias").

Value

The input `sens_results` object with `$cfi` updated.

Weight presets

balanced Equal weight to all available components (default).

rv_priority Double weight on both Robustness Value components. Use when the partial R-squared framework is most relevant.

evalue_priority Double weight on E-value. Use when risk-ratio framing is most natural (e.g. binary outcomes, relative risks).

itcv_priority Double weight on percent bias (ITCV/konfound). Use when case-replacement interpretability is prioritised.

Note

Presets are transparent defaults, not discipline-specific claims. Present chosen weights explicitly when reporting results.

Examples

```
if (requireNamespace("sensemakr", quietly = TRUE)) {
  data("darfur", package = "sensemakr")
  fit <- lm(peacefactor ~ directlyharmed + age + female, data = darfur)
  res <- run_sensitivity(fit, treatment = "directlyharmed", data = darfur)
  res <- compute_cfi_weighted(res, weights = "rv_priority")
  print_cfi(res)
}
```

---

detect_design *Detect the study design from a model object*

Description

Inspects a fitted model object and returns the most appropriate sensitivity analysis design type. The detected design determines which sensitivity frameworks are run by ‘run_sensitivity()‘.

Users can always override the detected design manually via the ‘design‘ argument in ‘sens_report()‘ or ‘run_sensitivity()‘.

Usage

```
detect_design(model, verbose = TRUE)
```

Arguments

model	A fitted model object. Supported classes: • ‘lm‘, ‘glm‘ → ‘"regression"‘ • ‘lmerMod‘, ‘glmerMod‘ (lme4) → ‘"regression"‘ • Objects with class ‘"Match"‘ (Matching package) → ‘"matched"‘ • ‘coxph‘, ‘survreg‘ (survival) → ‘"survival"‘ • ‘ivreg‘ (ivreg/AER) → ‘"iv"‘
verbose	Logical. If ‘TRUE‘, prints the detected design. Default ‘TRUE‘.

Value

A character string: one of ‘"regression"‘, ‘"matched"‘, ‘"survival"‘, ‘"iv"‘, or ‘"unknown"‘.

Examples

```
fit <- lm(mpg ~ am + wt + hp, data = mtcars)
detect_design(fit)
```

---

extract_sensemakr *Extract and standardise results from a sensemakr object*

Description

Inspects a fitted sensemakr object and returns a clean named list with correctly mapped field names, ready to pass to new_sens_results().

This function exists because sensemakr internal field names (rv_q, rv_qa) are not immediately obvious from the printed output, and differ across versions. Always use this extractor rather than accessing sm$sensitivity_stats directly.

Usage

```
extract_sensemakr(sm)
```

Arguments

sm	A sensemakr object returned by sensemakr::sensemakr().

Value

A named list with elements:

estimate Point estimate of the treatment effect.

se Standard error.

t_stat t-statistic.

r2yd_x Partial R-squared of treatment with outcome.

r2dz_x Partial R-squared of treatment with treatment (benchmark).

rv_q Robustness Value for q=1 (point-estimate nullification).

rv_qa Robustness Value for q=1, alpha=.05 (significance).

raw_object The original `sensemakr` object.

Examples

```
if (requireNamespace("sensemakr", quietly = TRUE)) {
  data("darfur", package = "sensemakr")
  fit <- lm(peacefactor ~ directlyharmed + age + female + village,
            data = darfur)
  sm  <- sensemakr::sensemakr(fit, treatment = "directlyharmed")
  extract_sensemakr(sm)
}
```

`flag_fragility` *Flag the fragility of a sensitivity analysis conclusion*

Description

Classifies the robustness of a causal conclusion using a five-level scale separately for (1) point-estimate nullification and (2) statistical significance, where supported by the framework.

Usage

```
flag_fragility(x, thresholds = NULL)
```

Arguments

`x`	A `sens_results` object from `run_sensitivity()`.
`thresholds`	Optional named list to override default thresholds.

Value

The input `sens_results` object with `$fragility` populated as a named list with elements `point`, `alpha`, `point_label`, `alpha_label`.

Classification scale

Level	RV threshold	Meaning
Highly stable	>= 0.20	Very strong confounding needed
Stable	0.10 – 0.20	Moderate confounding needed
Moderately fragile	0.05 – 0.10	Plausible confounding could matter
Fragile	0.01 – 0.05	Weak confounding could overturn result
Highly fragile	< 0.01	Minimal confounding overturns result

Examples

```
res <- new_sens_results(
  design = "regression", treatment = "t", outcome = "y",
  frameworks = "sensemakr",
  results = list(sensemakr = list(rv_q = 0.14, rv_qa = 0.08,
                                  r2yd_x = 0.02, estimate = 0.10))
)
res <- flag_fragility(res)
res$fragility
```

`generate_report` *Generate a structured sensitivity analysis report*

Description

Produces a complete, structured report from a `sens_results` object. The report includes study design, numeric results, fragility classification, narrative interpretation, and suggested reporting language for academic papers.

Works fully offline without an API key. If `interpret_sensitivity()` has not been run, the function runs it automatically using template-based narratives.

Usage

```
generate_report(
  x,
  format = c("markdown", "text"),
  include_citation = TRUE,
  verbosity = "standard",
  file = NULL
)
```

Arguments

x	A sens_results object, ideally with fragility assessed.
format	Character. Output format. One of "markdown", "text". Default "markdown".
include_citation	Logical. Append reference list. Default TRUE.
verbosity	Character. Narrative verbosity passed to interpret_sensitivity() if narrative not yet generated. Default "standard".
file	Character. Optional file path to write the report. If NULL (default), returns the report as a character string and prints it.

Value

Invisibly returns the report as a character string. If file is specified, also writes to disk.

Examples

```
res <- new_sens_results(
  design = "regression", treatment = "directlyharmed",
  outcome = "peacefactor", frameworks = c("sensemakr", "evalue"),
  results = list(
    sensemakr = list(estimate = 0.097, r2yd_x = 0.022,
                     rv_q = 0.139, rv_qa = 0.076),
    evalue    = list(estimate = 0.097, evalue = 1.90,
                     evalue_lower = 1.55)
  ),
  fragility = list(point = "Stable", alpha = "Moderately fragile",
                   point_label = "RV = 13.9%", alpha_label = "RV = 7.6%")
)
report <- generate_report(res)
cat(report)
```

---

interpret_sensitivity *Interpret sensitivity results using an LLM or concise templates*

Description

Generates a concise plain-language interpretation (2-3 sentences per framework) of sensitivity analysis findings. Uses an LLM if configured via use_llm_provider(); falls back to pre-written concise templates when no LLM is available.

The LLM assists only in phrasing. All statistical values come from the sens_results object computed by transparent R functions.

Usage

```
interpret_sensitivity(x, verbosity = "standard", cache = TRUE, ...)
```

Arguments

x	A sens_results object with fragility assessed.
verbosity	Character. Controls output length. "brief" = one sentence per framework. "standard" = 2-3 sentences per framework (default). "reviewer-ready" = full methods paragraph with citations.
cache	Logical. Cache the LLM response to avoid repeat API calls. Default TRUE.
...	Reserved for future use.

Value

The input sens_results object with $narrative populated.

Examples

```
res <- new_sens_results(
  design = "regression", treatment = "directlyharmed",
  outcome = "peacefactor", frameworks = "sensemakr",
  results = list(sensemakr = list(
    rv_q = 0.139, rv_qa = 0.076, r2yd_x = 0.022, estimate = 0.097
  )),
  fragility = list(point = "Stable", alpha = "Moderately fragile",
                   point_label = "", alpha_label = "")
)
res <- interpret_sensitivity(res)
cat(res$narrative)
```

---

is_sens_results *Check if object is a sens_results*

Description

Check if object is a sens_results

Usage

```
is_sens_results(x)
```

Arguments

x	Any R object.

Value

Logical.

new_sens_results *Create a sens_results object*

Description

Constructor for the unified sens_results S3 class. This is the central data structure that all causalfrag functions produce and consume.

Usage

```
new_sens_results(
  design,
  treatment,
  outcome,
  frameworks,
  results,
  fragility = NULL,
  narrative = NULL,
  llm_used = FALSE,
  call = NULL
)
```

Arguments

design	Character. Detected study design. One of "regression", "matched", "iv", "survival".
treatment	Character. Name of the treatment variable.
outcome	Character. Name of the outcome variable.
frameworks	Character vector. Which sensitivity frameworks were run.
results	Named list. Raw numeric results from each framework.
fragility	Named list with elements point and alpha.
narrative	Character. Plain-language interpretation.
llm_used	Logical. Whether an LLM was used for narrative generation.
call	The matched call.

Value

An object of class sens_results.

Examples

```
res <- new_sens_results(
  design    = "regression",
  treatment = "treat",
  outcome   = "outcome",
```

```
  frameworks = "sensemakr",
  results   = list(sensemakr = list(rv_q = 0.14, rv_qa = 0.08,
                                    r2yd_x = 0.02, estimate = 0.10))
)
print(res)
```

`plot.sens_results` *Plot method for sens_results*

Description

Plot method for sens_results

Usage

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

Arguments

`x`	A `sens_results` object.
`...`	Passed to `visualize_sensitivity()`.

Value

Invisibly returns `x`, the unchanged `sens_results` object. The method is called for its side effect of drawing a sensitivity plot.

`print.sens_results` *Print method for sens_results*

Description

Print method for sens_results

Usage

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

Arguments

`x`	A `sens_results` object.
`...`	Further arguments (unused).

Value

Invisibly returns `x`, the unchanged `sens_results` object. The method is called for its side effect of printing a formatted summary.

---

print_cfi *Print the Causal Fragility Index*

Description

Displays a formatted summary of the CFI score, component breakdown, and interpretation. Call after `compute_cfi()`.

Usage

```
print_cfi(x, ...)
```

Arguments

`x`	A `sens_results` object with `$cfi` populated.
`...`	Further arguments (unused).

Value

Invisibly returns `x`.

---

run_sensitivity *Run sensitivity analysis across frameworks*

Description

Dispatches to the appropriate sensitivity analysis packages based on the detected or user-specified study design. Returns a unified ‘sens_results‘ object containing numeric results from all frameworks run.

Each framework is called only if the required package is installed. Missing packages produce a warning, not an error, so the function always returns whatever results it can compute.

Usage

```
run_sensitivity(
  model,
  treatment,
  data,
  design = NULL,
  frameworks = NULL,
  benchmark_covariates = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

model	A fitted model object (e.g. from lm(), glm()).
treatment	Character. Name of the treatment variable.
data	A data frame containing the model variables.
design	Character. Study design. If NULL, calls detect_design() automatically. One of "regression", "matched", "iv", "survival".
frameworks	Character vector. Which frameworks to run. If NULL, selects automatically based on design. Subset of c("sensemakr", "evalue", "itcv", "rosenbaum").
benchmark_covariates	Character vector. Covariate names to use as benchmarks in sensemakr. Default NULL.
verbose	Logical. Print progress messages. Default TRUE.
...	Additional arguments (reserved for future use).

Value

A sens_results object with $results populated for each framework successfully run.

Examples

```
if (requireNamespace("sensemakr", quietly = TRUE)) {
  data("darfur", package = "sensemakr")
  fit <- lm(peacefactor ~ directlyharmed + age + female + village,
            data = darfur)
  res <- run_sensitivity(fit, treatment = "directlyharmed", data = darfur)
  print(res)
}
```

---

sens_report *Run the full sensitivity analysis pipeline*

Description

The main user-facing function. It runs the available sensitivity-analysis frameworks, classifies fragility, creates a narrative interpretation, and optionally draws a sensitivity plot when a supported plotting object is available.

Usage

```
sens_report(
  model,
  treatment,
  data,
```

```
  design = NULL,
  narrative = "standard",
  plot = TRUE,
  ...
)
```

Arguments

model	A fitted model object (for example, from ‘lm()‘ or ‘glm()‘).
treatment	Character. Name of the treatment variable in the model.
data	A data frame containing the variables in ‘model‘.
design	Character. Optional study-design override. One of ‘"regression"‘, ‘"matched"‘, ‘"iv"‘, or ‘"survival"‘. The default, ‘NULL‘, detects the design automatically.
narrative	Character. Narrative detail: ‘"brief"‘, ‘"standard"‘, or ‘"reviewer-ready"‘. The default is ‘"standard"‘.
plot	Logical. If ‘TRUE‘, draw a sensitivity plot when the required plotting object and package are available. The default is ‘TRUE‘.
...	Additional arguments passed to ‘run_sensitivity()‘.

Value

A ‘sens_results‘ object containing the framework-specific numeric results, fragility classifications, composite index when computable, and narrative interpretation. When ‘plot = TRUE‘, a plot may also be produced as a side effect.

Examples

```
if (requireNamespace("sensemakr", quietly = TRUE)) {
  data("darfur", package = "sensemakr")
  fit <- lm(peacefactor ~ directlyharmed + age + female + village,
            data = darfur)
  result <- sens_report(
    fit,
    treatment = "directlyharmed",
    data = darfur,
    frameworks = "sensemakr",
    plot = FALSE
  )
  print(result)
}
```

summary.sens_results *Summary method for sens_results*

Description

Summary method for sens_results

Usage

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

Arguments

object	A sens_results object.
...	Further arguments (unused).

Value

Invisibly returns object, the unchanged sens_results object. The method is called for its side effect of printing a concise summary of the study, frameworks, fragility assessment, and narrative status.

use_llm_provider *Configure the LLM provider for causalfrag*

Description

Sets up the LLM provider used by ‘interpret_sensitivity()‘ and ‘generate_report()‘. Configuration is stored in the R session environment.

The package works fully without calling this function. When no LLM is configured, narrative output falls back to pre-written templates based on the numeric results.

Usage

```
use_llm_provider(
  provider = c("openai", "anthropic", "none"),
  api_key = NULL,
  model = NULL,
  max_tokens = 512L
)
```

Arguments

provider	Character. One of '"openai"', '"anthropic"', '"none"'. Use '"none"' to explicitly disable LLM and use templates only.
api_key	Character. Your API key. If 'NULL', the function looks for environment variables 'OPENAI_API_KEY' or 'ANTHROPIC_API_KEY'.
model	Character. Model name to use. • OpenAI default: '"gpt-4o"' • Anthropic default: '"claude-3-5-haiku-latest"'
max_tokens	Integer. Maximum tokens for LLM response. Default 512.

Value

Invisibly returns a list of the current LLM configuration.

Examples

```
# Use Anthropic (key from environment variable)
use_llm_provider("anthropic")

# Use OpenAI with explicit key

use_llm_provider("openai", api_key = "sk-...")


# Disable LLM explicitly (template mode)
use_llm_provider("none")
```

---

visualize_sensitivity *Plot sensitivity contours from a sens_results object*

Description

Produces sensitivity analysis figures using **sensemakr**'s built-in plotting functions, or **confoundvis** if installed and `engine` is set accordingly. Plots are generated for each framework in the `sens_results` object that supports visualisation.

Usage

```
visualize_sensitivity(x, engine = "sensemakr", type = "contour", ...)
```

Arguments

x	A sens_results object.
engine	Character. One of "sensemakr", "confoundvis". Default "sensemakr" (uses the raw sensemakr object stored in x$results$sensemakr$raw_object).
type	Character. Plot type for sensemakr engine. One of "contour", "extreme". Default "contour".
...	Additional arguments passed to the plotting function.

Value

Invisibly returns x. Called for side effects (plots).

Examples

```
if (requireNamespace("sensemakr", quietly = TRUE)) {
  data("darfur", package = "sensemakr")
  fit <- lm(peacefactor ~ directlyharmed + age + female + village,
            data = darfur)
  res <- run_sensitivity(fit, treatment = "directlyharmed", data = darfur,
                         benchmark_covariates = "female")
  visualize_sensitivity(res)
  visualize_sensitivity(res, type = "extreme")
}
```

Index