Package: regsensitivity (via r-universe)

October 20, 2026

Type Package

Title Regression Sensitivity Analysis for Omitted Variable Bias

Version 0.2.0

Description Sensitivity analysis for linear regression coefficients in the presence of omitted variables. Implements the identified-set and breakdown-point analyses of Diegert, Masten and Poirier (2026) <doi:10.48550/arXiv.2206.02303>, Oster (2019) <doi:10.1080/07350015.2016.1227711> and Masten and Poirier (2026) <doi:10.1257/aer.20230242>. Provides bounds on regression coefficients under relaxations of the no-omitted-variables assumption indexed by interpretable sensitivity parameters, the breakdown point of a hypothesis about the long-regression coefficient, and plotting helpers.

License MIT + file LICENSE

URL https://github.com/mikenguyen13/regsensitivity, https://mikenguyen13.github.io/regsensitivity/

BugReports https://github.com/mikenguyen13/regsensitivity/issues

Encoding UTF-8

Language en-US

LazyData true

Depends R (>= 4.1)

Imports stats, grid, parallel, nloptr (>= 2.0.0), ggplot2 (>= 3.5.0), rlang (>= 1.0.0)

Suggests generics, shiny, haven (>= 2.5.0), AER, sensemakr, testthat (>= 3.0.0), knitr (>= 1.40), rmarkdown (>= 2.20), pdftools, covr, pkgdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/Needs/coverage covr

Config/Needs/lint lintr

RoxygenNote 7.3.3

NeedsCompilation no

Author Mike Nguyen [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-3432-8595>)

Maintainer Mike Nguyen <nguyennghia1301@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-20 13:50:02 UTC

RemoteUrl https://github.com/cran/regsensitivity

RemoteRef HEAD

RemoteSha 7b8985efb6094db7f2770016baf66b67ab5e0e83

Contents

as.data.frame.regsensitivity . . . 2
autoplot.regsensitivity . . . 4
bfg2020 . . . 4
calibrate_partial_r2 . . . 5
calibrate_rho . . . 6
hypothesis_helpers . . . 7
plot.regsensitivity . . . 8
regsen_boot . . . 9
regsen_bounds . . . 11
regsen_breakdown . . . 13
regsen_cores . . . 16
regsen_explore . . . 17
regsen_multi . . . 18
regsen_summary . . . 19
regsen_table . . . 20
regsen_tidy . . . 22
regsensitivity . . . 23
scale_colour_regsen . . . 24
theme_regsen . . . 24

Index **26**

`as.data.frame.regsensitivity`

Extract sensitivity results as a tidy data frame

Description

Returns the results table of a `regsensitivity` object as a plain `data.frame`, with the sensitivity parameters and the bounds (or the breakdown frontier) in columns.

Usage

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

Arguments

`x`	A `regsensitivity` object, from `regsen_bounds()` or `regsen_breakdown()`.
`row.names, optional`	Ignored, present for S3 compatibility with `base::as.data.frame()`.
`...`	Ignored.
`digits`	Optional number of significant digits to round numeric columns to. `NULL` (default) leaves the values at full precision, which is what you want when the result feeds further computation.
`finite_only`	Logical. If `TRUE`, drop rows where every bound is non-finite. Defaults to `FALSE` so that the unbounded region of the sweep stays visible.

Details

This is the recommended entry point for producing tables. Because the return value is an ordinary data frame it can be handed to whichever table package you already use – `knitr::kable()`, **kableExtra**, **gt**, **modelsummary**, **huxtable**, **tinytable** or **xtable** – rather than committing the package to one of them. `regsen_table()` wraps the common case.

Value

A `data.frame`.

Examples

```
data(bfg2020)
res <- regsen_bounds(
    avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 + log_area_2010 + lat + lon,
    data = bfg2020, compare = c("log_area_2010", "lat", "lon"), cbar = 0.1
)
head(as.data.frame(res))
```

autoplot.regsensitivity *Autoplot method*

Description

ggplot2::autoplot() support, so a regsensitivity object can be drawn by code that dispatches on autoplot() rather than plot(). It forwards every argument to plot.regsensitivity().

Usage

```
## S3 method for class 'regsensitivity'
autoplot(object, ...)
```

Arguments

object	A regsensitivity object.
...	Passed to plot.regsensitivity().

Value

A ggplot object.

Examples

```
data(bfg2020)
res <- regsen_bounds(
    avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 + log_area_2010 + lat + lon,
    data = bfg2020, compare = c("log_area_2010", "lat", "lon"), cbar = 0.1
)
ggplot2::autoplot(res)
```

bfg2020 *Bazzi, Fiszbein and Gebresilasse (2020) "Frontier Culture" replication data*

Description

A subset of the data from Bazzi, Fiszbein and Gebresilasse (2020) used in the Diegert-Masten-Poirier (2026) sensitivity analysis vignette. Each row is a county; each column is a covariate used in the DMP empirical application.

Usage

```
bfg2020
```

Format

A data.frame with 2036 rows and 14 columns:

lat, lon Latitude and longitude.

statea State indicator (numeric).

avgrep2000to2016 Outcome: average Republican vote share, 2000-2016.

temp_mean, rain_mean, elev_mean Climate / topography.

d_coa, d_riv, d_lak Distance to coast / river / lake.

ave_gyi Average GYI.

tye_tfe890_500kNI_100_l6 Treatment: years of frontier exposure.

log_area_2010 Log area, 2010.

km_grid_cel_code Grid cell code used for clustering.

Source

Bazzi, Fiszbein and Gebresilasse (2020), "Frontier Culture: The Roots and Persistence of 'Rugged Individualism' in the United States", *Econometrica* doi:10.3982/ECTA16484. A 14-variable, 2036-row subset of their public replication data, used here to demonstrate the package.

Examples

```
data(bfg2020)
str(bfg2020)
```

---

calibrate_partial_r2 *Pairwise partial R-squared of calibration covariates*

Description

For each `W1[k]`, computes

$$R^2_{W_{1k} \sim W_{1,-k} \cdot W_0}$$

i.e. the R-squared of regressing the `W0`-residualized `W1[k]` on the `W0`-residualized `W1[-k]`. Matches Table 3 in DMP (2026): a quick read on how collinear the comparison controls are.

Usage

```
calibrate_partial_r2(
  formula,
  data,
  compare = NULL,
  nocompare = NULL,
  subset = NULL
)
```

Arguments

`formula`	Two-sided formula: `y ~ x + w1 + w2 + ...`. The first right-hand-side variable is the primary independent variable; the rest are controls.
`data`	A data.frame.
`compare`	Optional character vector of variables to use as the comparison set. Defaults to all controls if neither `compare` nor `nocompare` is given.
`nocompare`	Optional character vector of controls to *exclude* from the comparison set.
`subset`	Optional logical or integer vector indicating which rows of `data` to include in the estimation.

Value

A data.frame with columns `variable` and `R2`.

Examples

```
data(bfg2020)
calibrate_partial_r2(
  avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 +
    log_area_2010 + lat + lon,
  data = bfg2020
)
```

---

calibrate_rho — *Calibration parameters rho_k for rxbar*

Description

For each calibration covariate `W1[k]`, computes

$$\rho_k = \frac{\sqrt{\mathrm{Var}(\pi_{1,med,k} W_{1k})}}{\sqrt{\mathrm{Var}(\pi'_{1,med,-k} W_{1,-k})}},$$

where pi(1,med) is the OLS coefficient on W1 from the regression of the treatment X on (W0, W1). Variances are computed on the W0-residualized variables, matching the construction in DMP (2026) equation (3.5).

Usage

```
calibrate_rho(formula, data, compare = NULL, nocompare = NULL, subset = NULL)
```

Arguments

`formula`	Two-sided formula: `y ~ x + w1 + w2 + ...`. The first right-hand-side variable is the primary independent variable; the rest are controls.
`data`	A data.frame.
`compare`	Optional character vector of variables to use as the comparison set. Defaults to all controls if neither `compare` nor `nocompare` is given.
`nocompare`	Optional character vector of controls to *exclude* from the comparison set.
`subset`	Optional logical or integer vector indicating which rows of `data` to include in the estimation.

Details

These are point-identified reference values to compare the rxbar breakdown point against. See DMP (2026) section 3.4 and Table 4.

Value

A data.frame with columns `variable` and `rho` (a percentage).

Examples

```
data(bfg2020)
bfg2020$statea <- factor(bfg2020$statea)
w1 <- c("log_area_2010", "lat", "lon", "temp_mean", "rain_mean",
        "elev_mean", "d_coa", "d_riv", "d_lak", "ave_gyi")
form <- reformulate(c("tye_tfe890_500kNI_100_l6", w1, "statea"),
                    response = "avgrep2000to2016")
calibrate_rho(form, bfg2020, compare = w1)
```

`hypothesis_helpers` *Hypothesis-direction helpers*

Description

Convenience wrappers for specifying the direction of a hypothesis used by `regsen_breakdown()` and `regsen_bounds()`.

Usage

```
bnd_lb(x)

bnd_ub(x)

bnd_eq(x)
```

Arguments

x	Numeric scalar or vector of hypothesis values.

Value

A numeric vector of the same length as x, carrying a "sign" attribute – ">", "<" or "=" – that tells regsen_breakdown() and regsen_bounds() which direction of hypothesis the values state.

Examples

```
bnd_lb(0)
bnd_ub(4)
bnd_eq(0)
```

---

plot.regsensitivity *Plot a regression sensitivity analysis*

Description

Produces a ggplot object visualizing either an identified-set sweep (from regsen_bounds()) or a breakdown frontier (from regsen_breakdown()).

Usage

```
## S3 method for class 'regsensitivity'
plot(
  x,
  ywidth = NULL,
  ylim = NULL,
  show_breakdown = TRUE,
  show_legend = TRUE,
  xline = NULL,
  xline_colour = "grey40",
  xline_linetype = "dashed",
  xline_linewidth = 0.35,
  title = NULL,
  subtitle = NULL,
  xtitle = NULL,
  ytitle = NULL,
  base_size = 11,
  ...
)
```

Arguments

`x`	A `regsensitivity` object.
`ywidth`	Half-width of the y-axis in standard deviations of X. Ignored when `ylim` is set.
`ylim`	Optional two-element numeric vector giving the y-axis limits.
`show_breakdown`	Logical; draw a horizontal line at the hypothesis value. Defaults to `TRUE`.
`show_legend`	Logical; show the legend (only relevant when plotting bounds with multiple values of a second sensitivity parameter).
`xline`	Optional numeric vector of x-axis positions at which to draw reference lines, the equivalent of Stata's `xline()`. Useful for marking a value of `rmax`, a breakdown point, or any other threshold being discussed in the text.
`xline_colour`, `xline_linetype`, `xline_linewidth`	Appearance of the `xline` reference lines.
`title`, `subtitle`, `xtitle`, `ytitle`	Plot annotations. `NULL` uses defaults; `NA` drops the annotation entirely, which is usually what you want for a figure whose caption already carries the description.
`base_size`	Base font size passed to `theme_regsen()`. Use `9` for a two-column journal figure rendered about 3.3 inches wide.
`...`	Ignored.

Value

A `ggplot` object. Because it is an ordinary ggplot, every default here can be overridden by adding scales, themes or annotations to it.

Examples

```
data(bfg2020)
res <- regsen_bounds(
    avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 + log_area_2010 + lat + lon,
    data = bfg2020, compare = c("log_area_2010", "lat", "lon"),
    cbar = c(0.1, 0.5, 1)
)
plot(res, xline = 1, base_size = 9)
```

`regsen_boot` *Bootstrap confidence interval for the breakdown point*

Description

Computes a non-parametric (or cluster) bootstrap confidence interval for the breakdown point returned by `regsen_breakdown()` or the scalar `$breakdown` field of `regsen_bounds()`. Both a bias-corrected and accelerated (BCa) interval and a percentile interval are returned; `type` chooses which one `$ci` reports.

Usage

```
regsen_boot(
  formula,
  data,
  ...,
  type = c("bca", "perc"),
  R = 999L,
  cluster = NULL,
  level = 0.95,
  seed = NULL,
  ncores = NULL,
  show_progress = interactive()
)
```

Arguments

`formula`	Two-sided formula: `y ~ x + w1 + w2 + ...`. The first right-hand-side variable is the primary independent variable; the rest are controls.
`data`	A data.frame.
`...`	Additional arguments forwarded to `regsen_breakdown()` (the analysis to bootstrap).
`type`	Interval type reported in `$ci`: `"bca"` (default) or `"perc"`. Under `"bca"` both intervals are computed and stored; under `"perc"` the jackknife BCa needs is skipped and `$ci_bca` is `NA`. See Details.
`R`	Integer. Number of bootstrap replications. Defaults to 999.
`cluster`	Optional character scalar naming a column of `data` to resample at the cluster level (e.g. `"km_grid_cel_code"` for the BFG 2020 application). When NULL the standard non-parametric bootstrap is used.
`level`	Two-sided confidence level for the percentile CI. Default 0.95.
`seed`	Optional integer seed for reproducibility. Results are identical for a given `seed` regardless of `ncores`: each replicate draws its own seed from a vector generated once up front, so nothing depends on how the work was divided. Passing a `seed` leaves the session's random stream where it was, so a bootstrap run inside a larger simulation does not disturb that simulation's own draws. With `seed = NULL` the replicate seeds come from the session stream, which is advanced by that one draw and nothing else.
`ncores`	Number of cores for the replications and the jackknife. `NULL` (default) uses the session setting of `regsen_cores()`, which is 1 unless changed; `"auto"` uses all but two of the machine's cores. A progress bar is not shown when running in parallel. Capped at `R`, and at 2 while `R CMD check --as-cran` is running, which forbids more; results do not depend on the cap. The breakdown computation inside each replicate always runs serially, so cores are not oversubscribed.
`show_progress`	Logical; print progress bar.

Details

For DMP analyses the breakdown point is computed exactly as in `regsen_breakdown()`; when `rxbar`, `rybar` and `cbar` are all scalar the returned breakdown is the rxbar breakdown for the (scalar) hypothesis on beta. For Oster analyses the breakdown is the |deltal value at which the hypothesis first fails.

The BCa interval takes the percentile interval and shifts the quantiles it reads by two constants: `z0`, a median-bias correction equal to the normal quantile of the share of replicates below the point estimate, and `acceleration`, computed from the delete-one jackknife over sampling units – rows, or clusters when `cluster` is given. The jackknife costs one breakdown computation per unit, which under an i.i.d. bootstrap means one per row and usually dominates the run; both it and the replicates honour `ncores`, and `type = "perc"` skips it. When the jackknife is degenerate (every unit gives the same estimate, so the acceleration is undefined) the BCa interval falls back to the percentile interval and `acceleration` is `NA`.

Value

An object of class `regsensitivity_boot` containing: `point`, `replicates`, `ci` (the interval named by `type`), `ci_bca`, `ci_perc`, `z0`, `acceleration`, `type`, `level`, `R`, `cluster`, `na` (the number of replicates that could not be computed) and `infinite` (the number on which the hypothesis survived every value of the sensitivity parameter; these count as `+Inf` in the intervals rather than being dropped).

Examples

```
data(bfg2020)
bfg2020$statea <- factor(bfg2020$statea)
w1 <- c("log_area_2010", "lat", "lon", "temp_mean", "rain_mean",
        "elev_mean", "d_coa", "d_riv", "d_lak", "ave_gyi")
form <- reformulate(c("tye_tfe890_500kNI_100_l6", w1, "statea"),
                    response = "avgrep2000to2016")
set.seed(1)
bb <- regsen_boot(form, bfg2020, compare = w1, cbar = 1,
                  R = 199, cluster = "km_grid_cel_code")
print(bb)
```

`regsen_bounds` *Bounds on a regression coefficient under omitted-variable bias*

Description

Computes the identified set for the coefficient on the primary independent variable in the infeasible long regression, across a grid of sensitivity parameters. Implements the analyses of Diegert, Masten & Poirier (2026) (the default) and of Oster (2019) extended by Masten & Poirier (2026).

Usage

```
regsen_bounds(
  formula,
  data,
  analysis = c("dmp", "oster"),
  compare = NULL,
  nocompare = NULL,
  rxbar = NULL,
  rybar = Inf,
  cbar = 1,
  clow = 0,
  rybar_expr = NULL,
  delta = NULL,
  r2long = 1,
  maxovb = NA,
  delta_type = c("eq", "bound"),
  r2long_type = c("eq", "relative"),
  maxovb_type = c("bound", "relative"),
  beta = "sign",
  product = TRUE,
  subset = NULL,
  ncores = NULL
)
```

Arguments

`formula` Two-sided formula: `y ~ x + w1 + w2 + ...`. The first right-hand-side variable is the primary independent variable; the rest are controls.

`data` A data.frame.

`analysis` Which sensitivity analysis to run: `"dmp"` (default) or `"oster"`.

`compare` Optional character vector of variables to use as the comparison set. Defaults to all controls if neither `compare` nor `nocompare` is given.

`nocompare` Optional character vector of controls to *exclude* from the comparison set.

`rxbar`, `rybar`, `cbar`
(DMP) Numeric vectors of sensitivity-parameter values to sweep over. `rybar = Inf` (the default) gives the no-rybar case; setting it finite invokes the global-optimization code path.

`clow` (DMP) Lower bound on control endogeneity, the `clow` of DMP Assumption A6 `R(W2 ~ W1 . W0) %in% [clow, cbar]`. Default 0, which asserts nothing beyond `cbar`. A positive value asserts that the controls are *at least* that endogenous. Must satisfy `0 <= clow <= min(cbar)`.

`rybar_expr` (DMP) A function `function(rxbar) rybar` to set rybar as a function of rxbar (the only supported form in the Stata source is `rybar = rxbar`, i.e. `function(rxbar) rxbar`).

`delta`, `r2long`, `maxovb`
(Oster) Numeric vectors of sensitivity values.

delta_type	One of "eq" (equality, the default) or "bound".
r2long_type	One of "eq" (the default) or "relative". When "relative", values are multiplied by R-squared(medium).
maxovb_type	One of "bound" (default) or "relative". When "relative", values are multiplied by |Beta(medium)|.
beta	Hypothesis spec for the breakdown point. See regsen_breakdown().
product	Logical. If TRUE (default), all combinations of the sensitivity-parameter grids are evaluated; if FALSE, the inputs are zipped element-wise. Maps to Stata's noproduct option (inverted).
subset	Optional logical or integer vector indicating which rows of data to include in the estimation.
ncores	Number of cores to spread the grid over. NULL (default) uses the session setting of regsen_cores(), which is 1 unless changed; "auto" uses all but two of the machine's cores. Only the finite-rybar regime is costly enough to benefit. Results are identical for any number of cores.

Value

A regsensitivity object. The results field holds a data.frame with one row per sensitivity-parameter point.

For a breakdown analysis, results$breakdown is **signed**: its sign carries the direction of selection, and for Oster it is the delta that solves Proposition 3 for the hypothesised value. Feeding that value back into regsen_bounds() recovers the hypothesised beta, but feeding abs() of it lands on a different branch of the cubic. The scalar $breakdown field and the print method report the magnitude, since that is what is quoted as "the breakdown point". Note that the scalar exists on regsen_bounds() output only; a regsen_breakdown() result carries the value in results$breakdown alone.

Examples

```
data(bfg2020)
bnds <- regsen_bounds(
  avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 +
    log_area_2010 + lat + lon + temp_mean + rain_mean + elev_mean +
    d_coa + d_riv + d_lak + ave_gyi,
  data = bfg2020,
  cbar = 0.1
)
print(bnds)
```

---

regsen_breakdown *Breakdown frontier for a regression coefficient hypothesis*

Description

Find the smallest sensitivity-parameter value at which a given hypothesis about the long-regression coefficient first fails. For DMP, this is rxbar as a function of (cbar, rybar, beta) or – with `direction = "rybar"` – rybar as a function of (rxbar, cbar, beta). For Oster, this is |deltal| as a function of R-squared(long), beta and (optionally) maxovb.

Usage

```
regsen_breakdown(
  formula,
  data,
  analysis = c("dmp", "oster"),
  compare = NULL,
  nocompare = NULL,
  cbar = 1,
  clow = 0,
  rybar = Inf,
  rybar_expr = NULL,
  direction = c("rxbar", "rybar"),
  rxbar = NULL,
  r2long = 1,
  maxovb = NA,
  r2long_type = c("eq", "relative"),
  maxovb_type = c("bound", "relative"),
  beta = "sign",
  subset = NULL,
  ncores = NULL
)
```

Arguments

`formula` Two-sided formula: `y ~ x + w1 + w2 + ...`. The first right-hand-side variable is the primary independent variable; the rest are controls.

`data` A data.frame.

`analysis` Which sensitivity analysis to run: `"dmp"` (default) or `"oster"`.

`compare` Optional character vector of variables to use as the comparison set. Defaults to all controls if neither `compare` nor `nocompare` is given.

`nocompare` Optional character vector of controls to *exclude* from the comparison set.

`cbar`, `clow`, `rybar`, `rybar_expr`
(DMP) Same as in `regsen_bounds()`.

`direction` (DMP) Which sensitivity parameter the breakdown point is reported in: `"rxbar"` (default) sweeps `cbar` or `beta` and solves for rxbar; `"rybar"` sweeps `rxbar` and solves for rybar, the frontier `rybar_bf(rxbar)` of DMP (2026) Theorem 4. The two trace the same frontier, but only the rybar direction can describe its horizontal arm, where the conclusion survives every rxbar and the rxbar breakdown point is `+Inf`.

rxbar	(DMP, direction = "rybar") Numeric vector of rxbar values at which to evaluate the frontier. Defaults to an 11-point grid over [0, 2 * rmax(cbar)].
r2long, maxovb	(Oster) Same as in regsen_bounds().
r2long_type	One of "eq" (the default) or "relative". When "relative", values are multiplied by R-squared(medium).
maxovb_type	One of "bound" (default) or "relative". When "relative", values are multiplied by |Beta(medium)|.
beta	Hypothesis spec. One of: • "sign" – the hypothesis that sign(beta_long) = sign(beta_med). • a numeric scalar or vector. Use the helpers bnd_lb(), bnd_ub(), bnd_eq() to set the direction, e.g. beta = bnd_lb(0) for the hypothesis beta > 0.
subset	Optional logical or integer vector indicating which rows of data to include in the estimation.
ncores	Number of cores to spread the frontier's values over. NULL (default) uses the session setting of regsen_cores(); "auto" uses all but two of the machine's cores. Only frontiers that need the optimizer (finite rybar, rybar_expr, or direction = "rybar") are costly enough to benefit. Results are identical for any number of cores.

Details

For analysis = "oster" the hypothesis selects between two quantities that Masten and Poirier (2026) distinguish. An equality hypothesis (beta = bnd_eq(0)) returns the *explain away* breakdown point: the signed delta at which beta_long equals the hypothesized value, the number psacalc reports. An inequality or sign hypothesis (the default) returns the *sign change* breakdown point: the smallest |delta| at which some value on the wrong side of the hypothesis enters the identified set. The two can differ by an order of magnitude and the second is the one that bears on whether the conclusion could be wrong.

Their Theorem 2 shows the sign change breakdown point can never exceed one, so without maxovb the reported value is capped at 1: a printed 1 means the sign survives every |delta| below the conventional cutoff, not that a solution was found there. Supplying maxovb adds the assumption that lifts the cap.

Value

A regsensitivity object. results$index holds the swept parameter and results$breakdown the breakdown point at each value.

Examples

```
data(bfg2020)
bk <- regsen_breakdown(
  avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 +
    log_area_2010 + lat + lon + temp_mean + rain_mean + elev_mean +
    d_coa + d_riv + d_lak + ave_gyi,
  data = bfg2020,
  cbar = seq(0, 1, 0.1)
```

```
)
print(bk)
```

`regsen_cores` *Cores used by the package*

Description

Sets or reads the number of cores that `regsen_bounds()`, `regsen_breakdown()`, `regsen_multi()` and `regsen_boot()` use by default. Each of those also takes an `ncores` argument for a single call; this sets the session-wide default those arguments fall back to, held in `options(regsensitivity.ncores)`.

Usage

```
regsen_cores(n = NULL)
```

Arguments

`n` One of:

- `NULL` (default): read the current setting without changing it.
- a positive integer: use that many cores.
- `"auto"`: use `parallel::detectCores() - 2`, and at least 1. Two are left free so the session, and whatever else the machine is doing, stay responsive.

Whatever is set, at most 2 are used while `R CMD check --as-cran` is running, which forbids more.

Details

Parallel runs return exactly what serial runs return. The identified set and the breakdown frontiers draw no random numbers, and the bootstrap seeds each replicate from a vector drawn once up front, so nothing depends on how the work was divided.

On macOS and Linux the work is forked with `parallel::mclapply()`, so the data and model need no copying. Windows has no fork; it gets a PSOCK cluster, which is slower to start and pays to ship the data once, so there the gain is only worth having for long jobs (a bootstrap, or a finite-`rybar` sweep with many grid points).

What is parallel: the grid points of a `regsen_bounds()` sweep (only the finite-`rybar` regime costs anything; the closed forms are cheap either way), the values of a `regsen_breakdown()` frontier, the treatments of `regsen_multi()`, and the replicates and jackknife of `regsen_boot()`. Work inside a replicate or a treatment is run serially, so cores are never oversubscribed by nesting.

Value

The number of cores in effect after the call, invisibly when `n` is given.

Examples

```
old <- getOption("regsensitivity.ncores")
regsen_cores()          # the current setting, 1 unless changed
regsen_cores(2)         # run the sweeps on two cores
regsen_cores(1)         # back to serial
options(regsensitivity.ncores = old)
```

---

regsen_explore *Explore a sensitivity analysis interactively*

Description

Opens a **shiny** application for moving the sensitivity parameters by hand and watching the identified set respond. It is a way of building intuition, and of finding the region worth reporting, before committing to a figure.

Usage

```
regsen_explore(formula, data, compare = NULL, ...)
```

Arguments

`formula`	A model formula, as for `regsen_bounds()`.
`data`	A data frame.
`compare`	Character vector of comparison covariates.
`...`	Passed to `shiny::runApp()`, e.g. `launch.browser` or `port`.

Details

shiny is only needed to run this, so it lives in `Suggests`: the rest of the package works without it.

Value

Invisibly `NULL`; called for the running application.

Examples

```
# Starts a Shiny app, so it only runs in an interactive session.
if (interactive()) {
    data(bfg2020)
    bfg2020$statea <- factor(bfg2020$statea)
    w1 <- c("log_area_2010", "lat", "lon")
    regsen_explore(
        avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 +
            log_area_2010 + lat + lon,
        data = bfg2020, compare = w1
    )
}
```

regsen_multi *Sensitivity analysis for several treatments at once*

Description

Runs regsen_breakdown() (or regsen_bounds()) once per candidate treatment and collects the breakdown points into one table. This is the applied question "which of my regressors would survive an unobservable, and which would not?", answered in a single call.

Usage

```
regsen_multi(
  formula,
  data,
  treatments,
  compare = NULL,
  fun = regsen_breakdown,
  ...,
  ncores = NULL
)

## S3 method for class 'regsensitivity_multi'
plot(x, base_size = 11, ...)
```

Arguments

formula	A model formula. Its right-hand side must contain every name in treatments. Which term appears first does not matter here, since each treatment is promoted in turn.
data	A data frame.
treatments	Character vector of variables to treat as the treatment in turn.
compare	Passed to the underlying analysis. A variable currently acting as the treatment is removed from compare for that row, since a variable cannot calibrate against itself.
fun	The analysis to run: regsen_breakdown() (default) or regsen_bounds().
...	Passed to fun, e.g. cbar, analysis, beta.
ncores	Number of cores to spread the treatments over. NULL (default) uses the session setting of regsen_cores(); "auto" uses all but two of the machine's cores. Each treatment's analysis then runs serially inside, so cores are not oversubscribed.
x	A regsensitivity_multi object.
base_size	Base font size, passed to theme_regsen().

Details

Each treatment is analysed in the *same* model: the variable of interest moves to the front of the formula and every other right-hand-side term stays a control, including the other candidate treatments. That is deliberate – dropping the others would change the specification and the breakdown points would no longer be comparable across rows.

Value

A `data.frame` of class `regsensitivity_multi`, one row per treatment, with columns `treatment`, `estimate` (the medium-regression coefficient), `breakdown`, and `n`. Rows where the analysis failed carry `NA` and an `error` column explaining why, rather than aborting the whole sweep.

Examples

```
data(bfg2020)
w1 <- c("log_area_2010", "lat", "lon")
regsen_multi(
    avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 + log_area_2010 + lat + lon,
    data = bfg2020,
    treatments = c("tye_tfe890_500kNI_100_l6", "lat"),
    compare = w1, cbar = 1
)
```

`regsen_summary` *Sensitivity summary (DMP bounds + Oster breakdown)*

Description

Runs the default sweep used by Stata's `regsensitivity` when no subcommand is given: a DMP bounds analysis and an Oster breakdown analysis at a few standard r2long values.

Usage

```
regsen_summary(formula, data, compare = NULL, nocompare = NULL, subset = NULL)
```

Arguments

`formula`	Two-sided formula: `y ~ x + w1 + w2 + ...`. The first right-hand-side variable is the primary independent variable; the rest are controls.
`data`	A data.frame.
`compare`	Optional character vector of variables to use as the comparison set. Defaults to all controls if neither `compare` nor `nocompare` is given.
`nocompare`	Optional character vector of controls to *exclude* from the comparison set.
`subset`	Optional logical or integer vector indicating which rows of `data` to include in the estimation.

Value

A list with elements `dmp_bounds` and `oster_breakdown`, each a `regsensitivity` object.

Examples

```
data(bfg2020)
s <- regsen_summary(
  avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 +
    log_area_2010 + lat + lon,
  data = bfg2020
)
print(s)
```

---

regsen_table — *Render a sensitivity analysis as a publication-ready table*

Description

Formats the results of `regsen_bounds()` or `regsen_breakdown()` for a manuscript. The `format` argument selects the output flavour, so the same call works in an Rmd/Qmd document knitting to PDF, to HTML, or to a Markdown target such as GitHub or Typst.

Usage

```
regsen_table(
  x,
  format = NULL,
  digits = 3,
  caption = NULL,
  label = NULL,
  notes = NULL,
  booktabs = TRUE,
  threeparttable = TRUE,
  align = NULL,
  col_names = NULL,
  escape = TRUE,
  max_rows = NULL,
  ...
)
```

Arguments

`x`	A `regsensitivity` object.
`format`	One of `"latex"`, `"html"`, `"markdown"`, `"pipe"` or `"simple"`. Defaults to `"latex"` when knitting to PDF and `"markdown"` otherwise.
`digits`	Significant digits for the numeric columns.
`caption`	Table caption. `NULL` for none.

`label`	LaTeX label, used as `\\label{tab:<label>}`. Ignored for non-LaTeX formats.
`notes`	Character vector of table notes, one element per line. Set to `NULL` for none, or `TRUE` to use an automatically generated note recording the analysis, hypothesis and sample size – which is the information a referee will look for.
`booktabs`	Logical; use `booktabs` rules in LaTeX output.
`threeparttable`	Logical; wrap LaTeX output and its notes in a `threeparttable` environment. Requires `\\usepackage{threeparttable}` in the document preamble.
`align`	Column alignment string, e.g. `"lrrr"`. `NULL` left-aligns character columns and right-aligns numeric ones.
`col_names`	Optional character vector of column headings. `NULL` uses tidied versions of the internal names.
`escape`	Logical; escape special characters. Set `FALSE` if you are supplying LaTeX markup in `col_names` or `caption`.
`max_rows`	Optional integer. If the sweep is longer than this, the table is thinned to `max_rows` evenly spaced rows and a note records that this happened. `NULL` keeps every row.
`...`	Passed to `knitr::kable()`.

Details

For LaTeX the default follows the convention most economics journals expect: `booktabs` rules, and table notes wrapped in a `threeparttable` environment so the note block is set to the width of the table rather than the width of the page. Both are switchable.

Anything this function does not expose can be done by taking `as.data.frame()` of the object and passing it to the table package of your choice; nothing here is a dead end.

Value

A `knitr_kable` object, which prints as the requested markup and renders directly in an Rmd/Qmd chunk.

Examples

```
data(bfg2020)
res <- regsen_bounds(
    avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 + log_area_2010 + lat + lon,
    data = bfg2020, compare = c("log_area_2010", "lat", "lon"),
    cbar = c(0.1, 0.5, 1)
)
regsen_table(res, format = "markdown", digits = 3)
regsen_table(res, format = "latex", notes = TRUE,
             caption = "Identified sets", label = "idset")
```

regsen_tidy — *Tidy a sensitivity analysis*

Description

A **broom**-style tidier. `tidy()` returns one row per point of the sensitivity sweep; `glance()` returns a one-row summary of the analysis as a whole.

Usage

```
regsen_tidy(x, conf.int = TRUE, ...)

regsen_glance(x, ...)
```

Arguments

`x`	A `regsensitivity` object.
`conf.int`	Ignored; present for signature compatibility with other tidiers. Sensitivity bounds are not confidence intervals – see `regsen_boot()` for sampling uncertainty in the breakdown point.
`...`	Ignored.

Details

Implementing these makes `regsensitivity` objects usable by anything that speaks the broom vocabulary – most usefully **modelsummary**, which is how many economists assemble their tables.

The methods are registered on **generics**' `tidy()` and `glance()` only when that package is installed, so it stays an optional dependency. Without it, call `regsen_tidy()` and `regsen_glance()` directly.

Value

`regsen_tidy()` returns a `data.frame` with the sensitivity parameters plus `conf.low`/`conf.high` holding the identified set, using broom's column names so downstream packages recognise them. `regsen_glance()` returns a one-row `data.frame`.

Examples

```
data(bfg2020)
res <- regsen_bounds(
    avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 + log_area_2010 + lat + lon,
    data = bfg2020, compare = c("log_area_2010", "lat", "lon"),
    cbar = c(0.1, 0.5)
)
head(regsen_tidy(res))
regsen_glance(res)
```

regsensitivity *Regression sensitivity analysis*

Description

Top-level dispatcher that mirrors the Stata `regsensitivity` command. For most users, calling `regsen_bounds()` or `regsen_breakdown()` directly is clearer.

Usage

```
regsensitivity(
  subcommand = c("bounds", "breakdown", "summary"),
  formula,
  data,
  ...
)
```

Arguments

`subcommand`	One of `"bounds"`, `"breakdown"`, `"summary"`.
`formula`	Two-sided formula: `y ~ x + w1 + w2 + ...`. The first right-hand-side variable is the primary independent variable; the rest are controls.
`data`	A data.frame.
`...`	Additional arguments forwarded to the underlying function.

Value

An object of class `regsensitivity`.

See Also

`regsen_bounds()`, `regsen_breakdown()`, `regsen_summary()`

Examples

```
data(bfg2020)
regsensitivity(
  "bounds",
  avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 +
    log_area_2010 + lat + lon,
  data = bfg2020,
  cbar = 0.1
)
```

scale_colour_regsen *Colourblind-safe scales for sensitivity plots*

Description

Discrete colour and fill scales using the Okabe-Ito palette, which stays legible under the common forms of colour vision deficiency and survives greyscale printing. These are applied by default in `plot.regsensitivity()`; they are exported so a plot can be rebuilt with the same colours.

Usage

```
scale_colour_regsen(...)

scale_color_regsen(...)

scale_fill_regsen(...)
```

Arguments

`...`	Passed to `ggplot2::discrete_scale()`.

Value

A **ggplot2** scale.

Examples

```
d <- data.frame(
  x = rep(1:3, 2),
  y = c(1, 2, 3, 2, 3, 4),
  series = rep(c("a", "b"), each = 3)
)
ggplot2::ggplot(d, ggplot2::aes(x, y, colour = series)) +
  ggplot2::geom_line() +
  scale_colour_regsen()
```

theme_regsen *Publication-ready theme for sensitivity plots*

Description

A **ggplot2** theme tuned for figures that will be dropped into a paper. It follows the conventions of the economics journals: an L of axis lines rather than a box, no grid, ticks outside the panel, no figure title (the caption carries it), and a legend along the top so the plot region keeps its aspect ratio when the figure is scaled down to one column.

Usage

```
theme_regsen(
  base_size = 11,
  base_family = "",
  grid = c("none", "y", "x", "both"),
  legend_position = "top"
)
```

Arguments

`base_size`	Base font size in points.
`base_family`	Base font family. The empty string uses the device default, which keeps figures reproducible across machines.
`grid`	One of `"none"` (default), `"y"`, `"x"` or `"both"`, choosing which faint grid lines to draw. The default is none, following the economics journals: a figure should carry ink only where it carries data. Use `"y"` when readers need to take values off the axis.
`legend_position`	Passed to `ggplot2::theme()`. Defaults to `"top"`.

Details

To match a LaTeX manuscript's body type, pass a serif family, e.g. `theme_regsen(base_family = "Times New Roman")`. Fonts are left to the caller because what is installed varies by machine, and a missing family would make figures unreproducible.

The default `base_size` of 11 assumes a figure rendered at roughly 6 inches wide. For a two-column journal figure, render at 3.3 inches and pass `base_size = 9`; the type will then sit at about 8pt on the page.

Value

A **ggplot2** theme object, which can be added to any plot.

Examples

```
data(bfg2020)
res <- regsen_bounds(
    avgrep2000to2016 ~ tye_tfe890_500kNI_100_l6 + log_area_2010 + lat + lon,
    data = bfg2020, compare = c("log_area_2010", "lat", "lon"), cbar = 0.1
)
plot(res) + theme_regsen(base_size = 9)
```

Index

∗ **datasets**
bfg2020, 4

as.data.frame(), *21*
as.data.frame.regsensitivity, 2
autoplot.regsensitivity, 4

base::as.data.frame(), *3*
bfg2020, 4
bnd_eq *(*hypothesis_helpers*)*, 7
bnd_eq(), *15*
bnd_lb *(*hypothesis_helpers*)*, 7
bnd_lb(), *15*
bnd_ub *(*hypothesis_helpers*)*, 7
bnd_ub(), *15*

calibrate_partial_r2, 5
calibrate_rho, 6

ggplot2::discrete_scale(), *24*
ggplot2::theme(), *25*

hypothesis_helpers, 7

knitr::kable(), *21*

parallel::mclapply(), *16*
plot.regsensitivity, 8
plot.regsensitivity(), *4*, *24*
plot.regsensitivity_multi *(*regsen_multi*)*, 18

regsen_boot, 9
regsen_boot(), *16*, *22*
regsen_bounds, 11
regsen_bounds(), *3*, *7–9*, *13–18*, *20*, *23*
regsen_breakdown, 13
regsen_breakdown(), *3*, *7–11*, *13*, *16*, *18*, *20*, *23*
regsen_cores, 16
regsen_cores(), *10*, *13*, *15*, *18*
regsen_explore, 17
regsen_glance *(*regsen_tidy*)*, 22
regsen_glance(), *22*
regsen_multi, 18
regsen_multi(), *16*
regsen_summary, 19
regsen_summary(), *23*
regsen_table, 20
regsen_table(), *3*
regsen_tidy, 22
regsen_tidy(), *22*
regsensitivity, 23

scale_color_regsen *(*scale_colour_regsen*)*, 24
scale_colour_regsen, 24
scale_fill_regsen *(*scale_colour_regsen*)*, 24
shiny::runApp(), *17*

theme_regsen, 24
theme_regsen(), *9*, *18*