

Package: SCAtools (via r-universe)

October 13, 2026

Type Package

Title Direction-Aware Sufficient Condition Analysis

Version 0.5.1

Date 2026-10-12

Description Tests whether a high or low level of a condition is enough to guarantee a high or low level of an outcome. Such a claim predicts that one corner of the condition-outcome scatter plot is empty of observations. The package estimates the boundary of that empty corner, its size as a share of the scope (the effect size), a permutation test against random pairing, and a threshold table giving the condition level that guarantees each outcome level, in actual units, percentages, percentiles or standard deviations. Each of the four directions (high or low condition, high or low outcome) is stated as a claim and mapped to its empty corner. Because a condition can rule out low outcomes while barely moving the average, results can be set beside ordinary least-squares and median regression slopes, showing whether the data hold an average effect, a sufficiency effect or both. Also provides plots, influential-case diagnostics, scope-sensitivity and resampling checks, random-data generation and power analysis for planning studies. Frontiers are estimated with the 'NCA' package.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports ggplot2 (>= 3.4.0), NCA (>= 5.0.2), quantreg, stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

URL <https://github.com/youngchanresearcher/SCAtools>

BugReports <https://github.com/youngchanresearcher/SCAtools/issues>

NeedsCompilation no

Config/roxygen2/version 8.1.1

Author Young Chan [aut, cre]

Maintainer Young Chan <youngchanresearcher@gmail.com>

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev

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

Date/Publication 2026-10-13 14:20:02 UTC

RemoteUrl <https://github.com/cran/SCAtools>

RemoteRef HEAD

RemoteSha calce7249ac676f8f633e18c0cf98a1b2b70eb63

Contents

SCAtools-package	3
sca	5
sca_analysis	6
sca_boundary_type	9
sca_check_direction	10
sca_compare	10
sca_corner	12
sca_corner_location	13
sca_corner_map	14
sca_direction	15
sca_empty_space	15
sca_extract	16
sca_four_configurations	17
sca_legacy_names	18
sca_locationshift	19
sca_magnitude	20
sca_normalize	22
sca_outliers	23
sca_output	24
sca_plot	25
sca_power	26
sca_powerplot	27
sca_random	28
sca_reference	29
sca_relationship	30
sca_resample	31
sca_rescale	33
sca_scales	34
sca_scope	34
sca_scope_sensitivity	35
sca_table	36
sca_terms	38

sca_test_plot	39
sca_thresholds	39
Index	42

SCAtools-package	<i>SCAtools: Direction-Aware Sufficient Condition Analysis</i>
------------------	--

Description

Tests whether a high or low level of a condition is enough to guarantee a high or low level of an outcome. Such a claim predicts that one corner of the condition-outcome scatter plot is empty of observations. The package estimates the boundary of that empty corner, its size as a share of the scope (the effect size), a permutation test against random pairing, and a threshold table giving the condition level that guarantees each outcome level, in actual units, percentages, percentiles or standard deviations. Each of the four directions (high or low condition, high or low outcome) is stated as a claim and mapped to its empty corner. Because a condition can rule out low outcomes while barely moving the average, results can be set beside ordinary least-squares and median regression slopes, showing whether the data hold an average effect, a sufficiency effect or both. Also provides plots, influential-case diagnostics, scope-sensitivity and resampling checks, random-data generation and power analysis for planning studies. Frontiers are estimated with the 'NCA' package.

The method

A condition is sufficient for an outcome when the condition guarantees the outcome. In a scatter plot of condition against outcome, such a claim rules out one corner: high X sufficient for high Y, for instance, leaves no cases with high X and low Y. Necessary condition analysis (NCA; Dul, 2016) estimates the empty corner of a necessity claim. By contraposition a sufficiency claim is a necessity claim about the opposite levels – high X sufficient for high Y is low X necessary for low Y – so its empty corner can be estimated with the same frontier techniques. SCAtools states the four sufficiency directions as claims, maps each to the corner it leaves empty, and reports the effect size, permutation test and sufficiency thresholds in the vocabulary of condition analysis in degree, following Chan (in press), *From Necessity to Sufficiency and Necessity-and-Sufficiency: Extending Condition Analysis in Degree*.

What it is for

- Finding the condition level that guarantees each outcome level, reported in actual units, percentages, percentiles or standard deviations ([sca_thresholds\(\)](#)).
- Measuring an effect that regression does not show: a condition can rule out low outcomes while barely moving the average. [sca_compare\(\)](#) reports the two side by side.
- Comparing several conditions, each with its own direction, in one table ([sca_table\(\)](#)).
- Planning a study: [sca_power\(\)](#) estimates the sample size a sufficiency effect of a given size needs.

How to use it

1. Fit the claim with `sca_analysis()`, naming the condition, the outcome and the direction, and the scope if theory gives one. Use `test.rep` for the permutation test; `sca()` runs 1000 permutations and draws the plot in one step.
2. Read the results with `sca_table()`, `sca_thresholds()` and `sca_plot()`; `sca_output()` prints and draws them together.
3. Set the result beside regression with `sca_compare()` and `sca_locationshift()`.
4. Check it with `sca_test_plot()`, `sca_scope()`, `sca_scope_sensitivity()`, `sca_resample()` and `sca_outliers()`.

Reading the result

A fitted model describes the empty-space pattern in the sample under the stated scope and frontier technique, and its thresholds are empirical rules for the population and design studied. Claims beyond the sample – that the condition causes the outcome, that future cases will follow the frontier, or that the pattern holds once common causes are controlled – rest on the research design. Each direction predicts a single empty corner, which assumes the relationship is monotone, so plot the data before choosing one.

Regression as a reference

An ordinary least-squares or median regression line can be drawn beside the frontier as a central-tendency reference. It summarises the average effect and is never a component of a sufficiency claim. Chan (in press), *Mean Effects Are Not Sufficiency Effects: How Sufficient Condition Analysis Complements Regression Analysis*, sets out why the two kinds of effect are reported side by side; `sca_compare()` and `sca_four_configurations` follow it.

The engine

Frontier estimation, effect sizes, threshold estimation, permutation tests, random-data generation and power analysis are delegated to the NCA package by Jan Dul and Govert Buijs. SCAtools contributes the mapping from a logical sufficiency direction to the physically empty corner, SCA-specific tables and labels, direction-aware reporting scales, and the shared vocabulary of condition analysis in degree that the NSCA package also uses.

Author(s)

Maintainer: Young Chan <youngchanresearcher@gmail.com>

Authors:

- Young Chan <youngchanresearcher@gmail.com>

References

Chan, Y. (in press). *From Necessity to Sufficiency and Necessity-and-Sufficiency: Extending Condition Analysis in Degree*. University of Leeds.

Chan, Y. (in press). *Mean Effects Are Not Sufficiency Effects: How Sufficient Condition Analysis Complements Regression Analysis*. University of Leeds.

Dul, J. (2016). Necessary Condition Analysis (NCA): Logic and methodology of "necessary but not sufficient" causality. *Organizational Research Methods*, 19(1), 10-52. doi:[10.1177/1094428115584005](https://doi.org/10.1177/1094428115584005)

See Also

Useful links:

- <https://github.com/youngchanresearcher/SCAtools>
- Report bugs at <https://github.com/youngchanresearcher/SCAtools/issues>

sca

Quick sufficient condition analysis

Description

This convenience wrapper is identical to `sca_analysis()` but marks the result for automatic plotting when it is printed interactively, and runs a permutation test by default.

Usage

```
sca(
  data,
  x,
  y,
  direction = "HH",
  ceilings = c("ce_fdh", "cr_fdh"),
  reference = NULL,
  test.rep = 1000,
  test.seed = NULL
)
```

Arguments

<code>data</code>	A data frame or object coercible to a data frame.
<code>x</code>	Columns containing one or more conditions, selected by name or position.
<code>y</code>	A single outcome column, selected by name or position.
<code>direction</code>	Sufficiency direction(s): "HH", "LH", "HL", or "LL". The first letter is the X level and the second is the Y level.
<code>ceilings</code>	Empty-space frontier techniques understood by <code>NCA::nca_analysis()</code> , such as "ce_fdh" and "cr_fdh". "ols" is not one of them; it is moved to reference with a warning.
<code>reference</code>	Central-tendency lines drawn beside the frontier for comparison, or NULL for none. "ols" is the ordinary least-squares regression of the outcome on the condition, fitted to all the data. It estimates no empty space, so it carries no effect size, no permutation test and no threshold table. See <code>sca_reference()</code> .
<code>test.rep</code>	Number of permutation resamples. Zero skips the test.

`test.seed` Optional seed for the permutation test. The caller's random stream is restored afterwards, so setting it does not reseed the session. The engine derives its whole sequence from one draw, so this fixes every reported p-value.

Details

`test.rep` defaults to zero in `sca_analysis()`, where every inferential choice is made deliberately, and to 1000 here, where a quick result should still carry statistical evidence. `NCA::nsca()` takes the same default for the same reason.

It runs that test whether or not you asked for one, so pass `test.seed` to pin the p-value. An unseeded permutation test draws from the clock and reports a different number on every run, which is not a number to put in a paper; the Settings block says which of the two you got, and says so again beneath the results when the p-values are unpinned.

Value

An object of class `sca_result`.

Examples

```
# sca() runs the default 1000 permutations, which takes longer than
# an example should; sca_analysis() runs none unless asked.
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca(iv[seq(1, nrow(iv), by = 5), ], x = "X", y = "Y",
           direction = "HH", test.seed = 1)
fit
```

sca_analysis	<i>Sufficient condition analysis</i>
--------------	--------------------------------------

Description

Fits empty-space frontiers for directional sufficiency statements. The logical statement is converted to the physically empty scatter-plot corner by contraposition and passed to `NCA::nca_analysis()`.

Usage

```
sca_analysis(
  data,
  x,
  y,
  direction = "HH",
  ceilings = c("ce_fdh", "cr_fdh"),
  reference = NULL,
  custom = NULL,
  scope = NULL,
  threshold.x = "percentage.range",
```

```

threshold.y = "percentage.range",
convention = c("absolute", "directional"),
steps = 10,
step.size = NULL,
cutoff = 0,
qr.tau = 0.95,
reference.tau = 0.5,
test.rep = 0,
test.p_confidence = 0.95,
test.p_threshold = 0.05,
test.seed = NULL,
relevance = NULL,
na.action = c("pairwise", "listwise")
)

```

Arguments

data	A data frame or object coercible to a data frame.
x	Columns containing one or more conditions, selected by name or position.
y	A single outcome column, selected by name or position.
direction	Sufficiency direction(s): "HH", "LH", "HL", or "LL". The first letter is the X level and the second is the Y level.
ceilings	Empty-space frontier techniques understood by <code>NCA::nca_analysis()</code> , such as "ce_fdh" and "cr_fdh". "ols" is not one of them; it is moved to reference with a warning.
reference	Central-tendency lines drawn beside the frontier for comparison, or NULL for none. "ols" is the ordinary least-squares regression of the outcome on the condition, fitted to all the data. It estimates no empty space, so it carries no effect size, no permutation test and no threshold table. See <code>sca_reference()</code> .
custom	Optional intercept-slope pairs for custom frontier lines.
scope	Optional theoretical scope in the form <code>c(xmin, xmax, ymin, ymax)</code> .
threshold.x, threshold.y	Reporting scales for the condition and outcome axes of the threshold table. One of "actual", "percentage.range", "percentage.max", "percentile", or "sd".
convention	"absolute" places 0 at the low end of each axis and lets the inequality carry the direction. "directional" places 0 at the least sufficient end, so low-level directions are mirrored and every rule reads as "more of the sufficient thing".
steps	Number of threshold steps, or an explicit vector of outcome levels expressed on the threshold.y scale.
step.size	Optional threshold step size on the threshold.y scale.
cutoff	How out-of-range threshold values are represented; passed to <code>NCA::nca_analysis()</code> .
qr.tau	Quantile used by the quantile-regression frontier, stated as the share of observations the line is meant to leave on the non-empty side. It is read that way in both directions: the engine fits tau when the empty space is an upper corner

and $1 - \tau$ when it is a lower one (corners 3 and 4), so the default 0.95 is the 95th-percentile line where the empty space lies above and the 5th-percentile line where it lies below. Sufficiency in the "HH" direction has its empty space at high X with low Y, which is corner 4, and therefore fits the 5th percentile: a floor, not a ceiling. Do not invert `qr.tau` by hand to compensate.

Two reported quantities behave differently for `qr` than for the other frontiers, and both follow from the line being placed by a quantile rather than by the observations at the edge of the cloud. `frontier_accuracy` counts the share of observations on the non-empty side, which quantile regression sets to `qr.tau` by construction, so reading 0.95 back as evidence that the frontier fits well is circular: it is the input. `fit` compares this frontier's empty area with CE-FDH's and the engine returns NA whenever this area is the larger of the two, which a quantile line often produces because it is entitled to cut through observations where an envelope is not. An NA fit for `qr` is an ordinary outcome rather than a failure.

<code>reference.tau</code>	Quantile for a "qr" reference line. Defaults to 0.5, the median line: the central-tendency summary that does not move when one observation does, which is the reason to draw it beside OLS rather than instead of it. Separate from <code>qr.tau</code> , and deliberately so. A <code>qr frontier</code> sits at the edge of the cloud and uses <code>qr.tau</code> ; a <code>qr reference</code> sits in the middle of it and uses this. Both can be asked for in one analysis, which the engine cannot express – it takes one <code>qr.tau</code> per call – so reference lines are computed by this package rather than requested from it.
<code>test.rep</code>	Number of permutation resamples. Zero skips the test.
<code>test.p_confidence</code>	Confidence level for permutation p-value accuracy.
<code>test.p_threshold</code>	Significance threshold shown in test output.
<code>test.seed</code>	Optional seed for the permutation test. The caller's random stream is restored afterwards, so setting it does not reseed the session. The engine derives its whole sequence from one draw, so this fixes every reported p-value.
<code>relevance</code>	Optional pre-specified practical-relevance threshold for the effect size. Support requires an effect at least this large as well as a significant test. NULL means no relevance criterion was pre-specified, and support then rests on significance alone, as <code>NSCA::nsca_analysis()</code> has always required for its two components.
<code>na.action</code>	How missing values are dropped. "pairwise" fits each condition on its own complete pairs; "listwise" drops any row with a missing value anywhere, so every condition is fitted on the same rows. Under either, both axes of a condition are bounded by the same rows.

Details

Threshold values are always requested from the engine in actual units and converted to the reporting scale by SCAtools, so that a single, stated convention governs every axis and every direction. See `sca_scales()` for the available scales and `sca_thresholds()` for the resulting table.

Value

An object of class `sca_result` containing the NCA engine results and SCA-specific direction meta-data.

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",
                  scope = c(0, 100, 0, 100))

fit

# Report the condition axis as percentiles instead
sca_thresholds(fit, scale = "percentile")

# sca_table() shows the same model with a permutation test added.
```

sca_boundary_type	<i>Whether the boundary of a corner is a ceiling or a floor</i>
-------------------	---

Description

Boundary type follows the empty corner and nothing else. A corner at the top means the frontier bounds the data from above, which is a ceiling; at the bottom it is a floor. For sufficiency this makes the boundary a floor whenever the outcome level is high and a ceiling whenever it is low, which is why the technique names CE-FDH and CR-FDH keep the word "ceiling" even where the fitted boundary is a floor.

Usage

```
sca_boundary_type(corner)
```

Arguments

corner Integer vector containing values from 1 through 4.

Value

A character vector: "ceiling" or "floor".

See Also

[sca_corner_location\(\)](#), [sca_terms\(\)](#)

Examples

```
sca_boundary_type(1:4)
```

sca_check_direction	<i>Validate a set of directions</i>
---------------------	-------------------------------------

Description

Accepts one direction for every condition, or a single direction to be applied to all of them, and refuses anything else by name.

Usage

```
sca_check_direction(direction, n = length(direction))
```

Arguments

direction	Character vector containing "HH", "LH", "HL" or "LL". Case and surrounding space are ignored.
n	Number of conditions the directions must cover.

Value

The validated directions, recycled to length n.

See Also

[sca_corner\(\)](#), [sca_corner_map\(\)](#)

Examples

```
sca_check_direction("hh ")
sca_check_direction("HH", n = 3)
```

sca_compare	<i>Average effect beside sufficiency in degree</i>
-------------	--

Description

Reports the ordinary least-squares coefficient and the sufficiency effect size for the same relationship, and classifies the pair into the four configurations of the condition analysis in degree framework.

Usage

```
sca_compare(
  model,
  x = NULL,
  ceiling = NULL,
  beta_cut = .sca_beta_cut_default,
  d_cut = NULL
)
```

Arguments

<code>model</code>	An object returned by <code>sca_analysis()</code> .
<code>x</code>	Optional condition names or positions. All conditions by default.
<code>ceiling</code>	Frontier technique. Defaults to the first estimated.
<code>beta_cut</code>	Absolute standardised coefficient at or above which the average effect counts as present.
<code>d_cut</code>	Sufficiency effect size at or above which the empty corner counts as present. Defaults to the model's relevance, then to 0.10.

Value

A data frame with one row per condition.

What is being compared

beta is the standardised regression coefficient: how far the *expected* outcome moves with the condition. `effect_size` is the share of the declared scope left empty in the corner the sufficiency claim forbids. Neither implies the other. A relationship can have a strong average effect and no empty corner, or a clean empty corner and almost no slope.

The four configurations

- I, both** The average effect and the empty corner both reach their cut. Two separate findings about one relationship.
- II, sufficiency only** A flat slope with a largely empty corner. Typically a level shift or a threshold: beyond some level of the condition the low-outcome cases stop occurring, without further gain above it.
- III, average effect only** A slope with counterexamples still present. The condition raises the expected outcome without being enough for it – the reading most often mistaken for sufficiency.
- IV, neither** No evidence for either reading.

Configurations II and IV are indistinguishable in a design that reports only a regression coefficient, because both have a flat slope. Separating them is what this table is for.

The cuts

beta_cut defaults to 0.30, the lower edge of the medium band in Cohen (1988) for a standardised coefficient. d_cut defaults to the relevance threshold given to `sca_analysis()` when one was pre-specified – the researcher has already stated their criterion and this function should not override it – and otherwise to 0.10, the lower edge of the medium band in Dul (2016). The two numbers differ because the two conventions do; each is the bottom of "medium" in its own tradition. Whichever cuts were used are returned as columns, so a table always carries the lines that produced its own classification.

These are conventions, not benchmarks calibrated for this design.

effect_size_null and effect_size_excess sit beside the classification for that reason. A condition can clear d_cut on an effect size that is mostly the envelope fitting sample extremes – a noise condition at $n = 70$ reached 0.104 against the conventional 0.10 with a floor of 0.057 – and the label would read "dominant" either way. The cut is left where the convention puts it; what the table adds is the number that qualifies it.

References

Chan, Y. (in press). *Mean Effects Are Not Sufficiency Effects: How Sufficient Condition Analysis Complements Regression Analysis*. University of Leeds.

Chan, Y. (in press). *From Necessity to Sufficiency and Necessity-and-Sufficiency: Extending Condition Analysis in Degree*. University of Leeds.

See Also

`sca_locationshift()`, `sca_reference()`, `sca_table()`

Examples

```
# Configuration II: a flat slope with a largely empty corner
ii <- subset(sca_four_configurations, configuration == "II")
fit <- sca_analysis(ii, x = "X", y = "Y", direction = "HH",
                  ceilings = "ce_fdh", scope = c(0, 100, 0, 100))
sca_compare(fit)
```

sca_corner

Convert a sufficiency direction to its physical empty corner

Description

The first letter of a direction refers to X and the second to Y. For example, "HH" means high X is sufficient for high Y. The returned corner is a geometric location in the original, unflipped scatter plot.

Usage

```
sca_corner(direction = c("HH", "LH", "HL", "LL"))
```

Arguments

direction Character vector containing "HH", "LH", "HL", or "LL".

Value

An integer vector of physical empty corners.

Examples

```
sca_corner("HH")  
sca_corner(c("HH", "LH", "HL", "LL"))
```

sca_corner_location	<i>Where a physical corner sits on the plot</i>
---------------------	---

Description

Where a physical corner sits on the plot

Usage

```
sca_corner_location(corner)
```

Arguments

corner Integer vector containing values from 1 through 4.

Value

A character vector: "upper-left", "upper-right", "lower-left" or "lower-right".

See Also

[sca_boundary_type\(\)](#), [sca_corner_map\(\)](#)

Examples

```
sca_corner_location(1:4)
```

sca_corner_map	<i>Complete mapping between SCA directions and physical empty corners</i>
----------------	---

Description

Reproduces the sufficiency half of Table 1 of the condition analysis in degree framework, and adds the corner numbering the frontier engine uses. `relationship` states the direction as a claim about the theorised X-Y relationship; `empty_space` names what the expected empty space would contain; `boundary_type` says whether the boundary separating it is a ceiling or a floor.

Usage

```
sca_corner_map()
```

Details

The direction is a claim about the theorised relationship, not about the shape of the estimated frontier, which may be a step function or a straight line under any of the four types. What the four types do assume is that the relationship is monotone, because that is what makes the expected empty space a single corner of the plot. Theory may instead posit a curvilinear or U-shaped relationship – a condition that is sufficient over part of its range and not over the rest – and then no one corner is the expected empty space, the four directions cannot express the claim, and an effect size computed for any of them is answering a question that was not asked. Those relationships are outside the scope of this package. Plot the data before choosing a direction.

Value

A data frame that distinguishes SCA's logical direction number from the physical empty corner used by the frontier engine.

References

Chan, Y. (in press). *From Necessity to Sufficiency and Necessity-and-Sufficiency: Extending Condition Analysis in Degree*. University of Leeds.

Examples

```
sca_corner_map()
```

sca_direction	<i>Convert a physical empty corner to its sufficiency direction</i>
---------------	---

Description

Convert a physical empty corner to its sufficiency direction

Usage

```
sca_direction(corner = 1:4)
```

Arguments

corner	Integer vector containing values from 1 through 4. Corner 1 is upper-left, 2 upper-right, 3 lower-left, and 4 lower-right.
--------	--

Value

A character vector of sufficiency directions.

Examples

```
sca_direction(4)
sca_direction(1:4)
```

sca_empty_space	<i>What an expected empty space would contain</i>
-----------------	---

Description

Fills the variable names into the Table 1 wording for a corner.

Usage

```
sca_empty_space(corner, x = "X", y = "Y")
```

Arguments

corner	Integer vector containing values from 1 through 4.
x, y	Names of the condition and the outcome.

Value

A character vector, one entry per corner.

See Also

[sca_relationship\(\)](#), [sca_corner_map\(\)](#)

Examples

```
sca_empty_space(4)
sca_empty_space(4, "Training", "Performance")
```

sca_extract

Extract one value from an SCA result

Description

Extract one value from an SCA result

Usage

```
sca_extract(model, x = NULL, ceiling = NULL, param = "Effect size")
```

Arguments

model	An object returned by sca_analysis() .
x	Condition name. Defaults to the first condition.
ceiling	Frontier method. Defaults to the first available method.
param	An NCA parameter name such as "Effect size", one of the SCA metadata fields "direction", "sca_number", "empty_corner" and "statement", one of the geometry fields "relationship", "empty_space" and "boundary_type", or a name retired in 0.4.0.

Value

The selected value.

See Also

[sca_legacy_names\(\)](#)

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",
  scope = c(0, 100, 0, 100))
sca_extract(fit, param = "Effect size")
sca_extract(fit, ceiling = "cr_fdh", param = "Effect size")
sca_extract(fit, param = "relationship")
```

sca_four_configurations

Four comparative demonstrations of mean effect against sufficiency effect

Description

The simulated datasets behind the four configurations of Chan (in press), *Mean Effects Are Not Sufficiency Effects: How Sufficient Condition Analysis Complements Regression Analysis*. Each holds 500 cases on a theoretical scope of 0 to 100 on both axes, and each was built so that one of the four readings of an X-Y relationship holds.

Usage

```
sca_four_configurations
```

Format

A data frame with 2000 rows and 3 columns:

configuration "I", "II", "III" or "IV".

X The condition, on a theoretical scope of 0 to 100.

Y The outcome, on a theoretical scope of 0 to 100.

Why these four

A regression coefficient and a sufficiency effect size answer different questions, and neither implies the other. These datasets make the four possible combinations concrete, in the paper's own numbering, which runs from weakest to strongest:

"I" Neither focal effect. No slope and no empty corner.

"II" Sufficiency-dominant. A flat slope with a largely empty corner: beyond some level of X the low-Y cases stop occurring. **A regression-only design reports this as configuration I** – both have a flat slope – which is the point the four demonstrations exist to make.

"III" Mean-dominant. A slope with counterexamples still present: X raises the expected outcome without being enough for it.

"IV" Both. A slope and an empty corner. Two findings about one relationship, not one finding stated twice.

The scope these numbers assume

The four configurations were simulated on a 0-100 box on both axes, and every effect size documented for them is an area over that box. Pass it to reproduce them:

```
fit <- sca_analysis(subset(sca_four_configurations, configuration == "IV"),
  x = "X", y = "Y", direction = "HH",
  scope = c(0, 100, 0, 100),
  ceilings = c("ce_fdh", "cr_fdh"),
  test.rep = 5000, test.seed = 1)
sca_compare(fit)
```

Fitting on the observed range instead gives materially smaller effect sizes for the same data. Neither number is a fault in the other, and neither kind of scope is the correct one: they are answers to different questions, and `NCA::nca_analysis()` itself defaults to the observed range. What the two numbers show is that the denominator is half of the statistic, which is why `sca_analysis()` records both the scope it was given and the scope it used, and why `sca_scope()` reports them side by side.

Source

Simulated for the four comparative demonstrations. The same data is shipped as plain text at `system.file("extdata", "sca_four_configurations.csv", package = "SCAtools")`, so it can be read without R. The original per-configuration CSV files and a base-R reproduction script, which re-implements the CE-FDH frontier, the effect size and the permutation test from their definitions rather than calling this package, accompany the paper.

References

Chan, Y. (in press). *Mean Effects Are Not Sufficiency Effects: How Sufficient Condition Analysis Complements Regression Analysis*. University of Leeds.

See Also

[sca_compare\(\)](#), [sca_locationshift\(\)](#)

Examples

```
str(sca_four_configurations)
table(sca_four_configurations$configuration)

# The two that a regression cannot tell apart
flat <- subset(sca_four_configurations, configuration %in% c("I", "II"))
vapply(split(flat, flat$configuration),
  function(d) unname(stats::coef(stats::lm(Y ~ X, d))[[2L]]),
  numeric(1L))
```

Description

0.4.0 adopts the vocabulary of the condition analysis in degree framework. The threshold a sufficiency analysis reports is a *sufficiency threshold*, and *boundary* is reserved for the theoretical line an empty space is separated by, so the argument controlling the strictness of a reported inequality was renamed to *inequality*. 0.5.0 replaces this package's own shorthand "empty zone" with the framework's "empty space", so `empty_zone_area` became `empty_space_area`.

Usage

```
sca_legacy_names()
```

Details

Retired column names remain reachable. `sca_table()` and `sca_thresholds()` append them as duplicate columns when called with `legacy = TRUE`, each an exact copy of the current column. `sca_extract()` no longer accepts them: it stops and names the replacement. The retired arguments `boundary`, `bottleneck.x` and `bottleneck.y` were removed in 0.5.0; use `inequality`, `threshold.x` and `threshold.y`.

Value

A named character vector: names are the retired names, values the replacements.

See Also

```
sca_terms()
```

Examples

```
sca_legacy_names()
```

sca_locationshift	<i>Is the frontier a level shift or a threshold?</i>
-------------------	--

Description

Places three slopes side by side for the same relationship: the ordinary least-squares slope, the median (quantile regression at $\tau = 0.5$) slope, and the slope of the fitted frontier.

Usage

```
sca_locationshift(model, x = NULL, ceiling = NULL, tolerance = 0.1)
```

Arguments

model	An object returned by <code>sca_analysis()</code> .
x	Optional condition names or positions.
ceiling	Frontier technique. Defaults to the first estimated.
tolerance	Relative spread among the three slopes at or below which they are reported as agreeing.

Details

When all three agree, the frontier is a *location shift*: it is the central tendency moved bodily to the edge of the data, so sufficiency analysis adds a level to what the regression already said and does not add a slope. When the frontier slope departs from the other two, the boundary has a shape of its own and the relationship is not a shift.

This is the natural follow-up to configuration II of `sca_compare()`, where the average slope is flat but the forbidden corner is empty.

Only frontier techniques that fit a line report a slope. A step frontier – `ce_fdh`, `ce_vrs` – has none by construction, so `frontier_slope` is NA for those and the comparison needs a regression-type frontier such as `cr_fdh`.

Value

A data frame with one row per condition.

See Also

`sca_compare()`

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",
                   ceilings = "cr_fdh", scope = c(0, 100, 0, 100))
sca_locationshift(fit)
```

sca_magnitude	<i>Conventional magnitude bands</i>
---------------	-------------------------------------

Description

The bands Dul (2016) proposes for an NCA effect size and Cohen (1988) for a standardised regression coefficient, and the label a given value falls in. Called with no value it returns the band table.

Usage

```
sca_magnitude(value = NULL, scale = c("effect_size", "beta"))
```

Arguments

value	Optional numeric vector. Omitted, the band table is returned. For scale = "beta" the absolute value is used, since a band describes a magnitude and not a direction.
scale	Which convention: "effect_size" for Dul's bands, "beta" for Cohen's.

Value

With value, a character vector of labels, NA where the value is not finite. Without it, a data frame of bands.

What these are, and are not

They are a **convention to cite**, in the way a reader of a regression coefficient cites Cohen. Reporting "d = 0.35, large by the conventional bands (Dul, 2016)" is a citation, and a normal one.

They are **not** calibrated for your design. An effect size is the share of the declared scope left empty, so it depends on the scope you declared, the frontier technique, and – because the frontier is fitted to the sample's own extremes – on the sample size. Two studies of the same relationship at n = 30 and n = 800 will not report the same number, and neither is wrong: each describes its own data inside its own scope. What does not follow is that the two can be read off one band table without comment.

The two conventions share their edges and shift their labels

This is worth stating plainly, because the coincidence is a trap.

value	as a coefficient (Cohen)	as an effect size (Dul)
0.15	small	medium
0.35	medium	large
0.60	large	very large

Both use .10, .30 and .50 as edges; the labels sit one band apart. It is why `sca_compare()` defaults `beta_cut` to 0.30 and `d_cut` to 0.10 – each is the lower edge of "medium" in its own tradition, and those are different numbers even though the tradition's edges are not.

Read a band beside the floor

`effect_size_null` from `sca_table()` is what random pairing alone produces at that sample size, scope and technique. A band says where a number sits on a conventional scale; the floor says how much of it the geometry would have produced anyway. They answer different questions and belong in the same sentence.

References

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Lawrence Erlbaum.

Dul, J. (2016). Necessary Condition Analysis (NCA): Logic and methodology of "necessary but not sufficient" causality. *Organizational Research Methods*, 19(1), 10-52. doi:10.1177/1094428115584005

See Also

[sca_table\(\)](#), [sca_compare\(\)](#)

Examples

```
sca_magnitude()
sca_magnitude(scale = "beta")
sca_magnitude(c(0.04, 0.18, 0.35, 0.62))
# The same values under the other convention, one band apart:
sca_magnitude(c(0.04, 0.18, 0.35, 0.62), scale = "beta")
```

sca_normalize	<i>Normalize numeric variables for SCA</i>
---------------	--

Description

Normalize numeric variables for SCA

Usage

```
sca_normalize(data, scale = NULL, min_max = NULL)
```

Arguments

data	Input data frame or matrix.
scale	Output minimum-maximum pair, or one pair per column.
min_max	Optional empirical or theoretical minimum-maximum pair, or one pair per column.

Value

Data on the requested scales.

Examples

```
dat <- data.frame(X = c(0, 2, 4, 10), Y = c(10, 20, 30, 50))
sca_normalize(dat)
```

sca_outliers

*Diagnose observations that influence an SCA empty-space effect***Description**

Which observations hold up the frontier, and how far the effect size would move without them. A frontier is fixed by the most extreme cases rather than by the central tendency, so a handful of observations can carry a whole claim; this says how few.

Usage

```
sca_outliers(
  data,
  x = NULL,
  y = NULL,
  direction = "HH",
  ceiling = NULL,
  scope = NULL,
  k = 1,
  min.dif = 0.01,
  max.results = 25,
  condensed = FALSE
)
```

Arguments

data	A fitted <code>sca_analysis()</code> model, or a data frame.
x	With a model, optional condition names or positions; all conditions by default. With a data frame, one condition column selected by name or position.
y	One outcome column selected by name or position. Taken from the model when one is supplied, and an error if given as well.
direction	One sufficiency direction. Taken from the model when one is supplied, and an error if given as well.
ceiling	Frontier technique. CE-FDH is used when omitted; with a model, the first technique that model estimated.
scope	Optional theoretical scope. Taken from the model when one is supplied, and an error if given as well.
k	Number of observations considered jointly.
min.dif	Minimum relative effect-size difference.
max.results	Maximum number of rows returned.
condensed	Suppress redundant combinations when $k > 1$.

Value

An outlier-sensitivity table whose first column names the condition, or NULL when no observation reaches min.dif.

Two ways to call it

Pass a fitted model and the condition, outcome, direction and scope are taken from it, so the diagnostic cannot quietly describe a different analysis from the one being reported:

```
fit <- sca_analysis(data, x = c("A", "B"), y = "Y", direction = "HH")
sca_outliers(fit)
```

Every condition in the model is examined unless `x` selects some. Passing `y`, `direction` or `scope` alongside a model is an error rather than an override: the model fixes all three already, and a second answer for any of them would produce a table that does not describe the fit it came from.

Passing a data frame instead works as it always has, for the case where there is no fitted model to hand.

Reading it

This is a fragility check, not a cleaning step. An observation that moves the frontier is doing what the method says it should – frontiers are fixed by extreme cases – so finding one is not grounds for removing it. What the table says is how much of the claim rests on how few cases, which belongs in the write-up either way.

See Also

[sca_analysis\(\)](#), [sca_compare\(\)](#)

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv[seq(1, nrow(iv), by = 5), ], x = "X", y = "Y",
                  direction = "HH", ceilings = "ce_fdh",
                  scope = c(0, 100, 0, 100))
sca_outliers(fit)
```

sca_output

Display selected SCA output

Description

Display selected SCA output

Usage

```
sca_output(
  model,
  summaries = TRUE,
  thresholds = FALSE,
  plots = TRUE,
  tests = FALSE,
```

```
selection = NULL,  
ceiling = NULL,  
scale = NULL,  
outcome_scale = NULL,  
convention = NULL,  
inequality = c("strict", "inclusive")  
)
```

Arguments

model	An object returned by sca_analysis() .
summaries	Print the tidy summary table.
thresholds	Print sufficiency threshold rules.
plots	Draw frontier plots.
tests	Draw permutation-test plots when tests were requested.
selection	Optional condition names or positions.
ceiling	Frontier method used for threshold and permutation-test plots.
scale, outcome_scale, convention	Optional reporting overrides passed to sca_thresholds() . Each defaults to the choice made in sca_analysis() .
inequality	"strict" or "inclusive", passed to sca_thresholds() .

Value

Invisibly, a list containing generated plots and tables.

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")  
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",  
  ceilings = "ce_fdh", scope = c(0, 100, 0, 100))  
sca_output(fit, thresholds = TRUE)
```

sca_plot	<i>Plot SCA frontiers</i>
----------	---------------------------

Description

Plot SCA frontiers

Usage

```
sca_plot(model, x = NULL, ceilings = NULL, shade = TRUE, annotate = TRUE)
```

Arguments

model	An object returned by <code>sca_analysis()</code> .
x	Optional condition names or positions. All conditions are selected by default.
ceilings	Optional frontier methods to draw.
shade	Shade the empty zone under an exact CE-FDH or CE-VRS frontier.
annotate	Label the physical empty corner and logical direction. FALSE removes both labels, which is what a small multiple or a figure with its own caption wants.

Value

A ggplot for one condition or a named list of plots for multiple conditions.

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",
                   scope = c(0, 100, 0, 100), reference = "ols")
sca_plot(fit)
```

sca_power

Estimate power for a directional sufficiency analysis

Description

The frontier slope has to be able to bound the corner the direction implies. A rising line bounds the upper-left and lower-right corners, so it suits HH and LL; a falling line bounds the upper-right and lower-left corners, so it suits LH and HL. A horizontal line is accepted for every direction, as it is by the engine.

Usage

```
sca_power(
  n = c(20, 50, 100),
  effect = 0.1,
  slope = 1,
  ceiling = "ce_fdh",
  direction = "HH",
  p = 0.05,
  distribution.x = "uniform",
  distribution.y = "uniform",
  rep = 100,
  test.rep = 200,
  test.seed = NULL
)
```

Arguments

<code>n</code>	Sample size or vector of sample sizes.
<code>effect</code>	Population empty-space effect size(s).
<code>slope</code>	Frontier slope(s). The sign must match the direction: zero or positive for HH and LL, zero or negative for LH and HL.
<code>ceiling</code>	Frontier method(s).
<code>direction</code>	One sufficiency direction.
<code>p</code>	Significance level.
<code>distribution.x, distribution.y</code>	Marginal distribution(s).
<code>rep</code>	Number of simulated datasets per design cell.
<code>test.rep</code>	Number of permutations in each simulated analysis.
<code>test.seed</code>	Optional seed. A power table is the number that decides a sample size, so it has to reproduce; without one the engine draws from the clock and reports a different table on every run. The caller's random stream is restored afterwards, which matters more here than elsewhere because <code>NCA::nca_power()</code> calls <code>set.seed()</code> inside its own loop.

Value

A data frame of estimated power values with SCA direction metadata.

Examples

```
# Deliberately small: a reporting-grade run uses rep = 100 and
# test.rep = 200 and takes minutes rather than seconds.
power <- sca_power(n = c(20, 40), rep = 5, test.rep = 19, test.seed = 1)
power
```

sca_powerplot

Plot SCA power results

Description

Plot SCA power results

Usage

```
sca_powerplot(
  df_power,
  x.variable = "n",
  x.name = "Sample size",
  x.min = 0,
  x.max = NULL,
  line.variable = "ES",
  line.name = "Effect size"
)
```

Arguments

df_power	Data frame returned by <code>sca_power()</code> .
x.variable	Column mapped to the x axis.
x.name	X-axis label.
x.min, x.max	X-axis limits.
line.variable	Column mapped to line colour.
line.name	Legend title.

Value

A ggplot object.

Examples

```
power <- sca_power(n = c(20, 40), rep = 5, test.rep = 19, test.seed = 1)
sca_powerplot(power)
```

sca_random

Generate data with a directional sufficiency pattern

Description

Generate data with a directional sufficiency pattern

Usage

```
sca_random(
  n,
  intercepts,
  slopes,
  direction = "HH",
  distribution.x = "uniform",
  distribution.y = "uniform",
  mean.x = 0.5,
  mean.y = 0.5,
  sd.x = 0.2,
  sd.y = 0.2
)
```

Arguments

n	Number of observations.
intercepts	Intercept or vector of intercepts for frontier lines.
slopes	Slope or vector of slopes for frontier lines.
direction	One sufficiency direction: "HH", "LH", "HL", or "LL".

distribution.x, distribution.y Marginal distribution, "uniform" or "normal".
 mean.x, mean.y Means for truncated normal distributions.
 sd.x, sd.y Standard deviations for truncated normal distributions.

Value

A data frame with X condition columns and one Y outcome column.

Examples

```
set.seed(42)
dat <- sca_random(50, intercepts = 0, slopes = 1, direction = "HH")
head(dat)
```

sca_reference	<i>Central-tendency reference lines</i>
---------------	---

Description

Fits the central-tendency lines for each condition, on the same observations the frontier was fitted to. "ols" is the ordinary least-squares regression, the line $\text{lm}(y \sim x)$ returns. "qr" is a quantile regression at reference.tau, which defaults to 0.5: the median line, which is the central-tendency summary that does not move when one observation does.

Usage

```
sca_reference(model, x = NULL, line = NULL)
```

Arguments

model An object returned by [sca_analysis\(\)](#).
 x Optional condition names or positions. All conditions by default.
 line Which reference lines to report. Defaults to those the model asked for, or to "ols" when it asked for none.

Details

Reported separately from [sca_table\(\)](#) on purpose. Regression describes how the expected outcome moves with the condition; a sufficiency frontier describes which condition-outcome combinations are absent. Section 6.1 of the condition analysis in degree framework treats these as different targets that are most informative shown side by side rather than merged into one number, and section 4.3 notes that a frontier is fixed by the most extreme observations rather than by the central tendency. A reference line therefore has a slope and an intercept but no empty space, no effect size, no permutation p value and no sufficiency threshold.

"qr" as a reference is not the same thing as "qr" as a frontier, and they are set by different arguments. A frontier qr uses qr.tau, defaults to 0.95 and is meant to sit at the edge of the cloud; a

reference qr uses reference.tau, defaults to 0.5 and is meant to sit in the middle of it. Both can be asked for at once, which the engine could not do in a single call: it takes one qr.tau, so the reference is computed here instead.

r_squared and beta are NA for "qr", and pseudo_r1 is NA for "ols". Quantile regression minimises a different loss, so there is no residual variance to take a share of; Koenker and Machado's R1 is the right measure for it, and it is a share of absolute loss at one quantile rather than of variance. Reporting it under the r_squared heading would be one column carrying two meanings, and leaving it out would report a difference in kind as a missing value, so it has a column of its own. A pseudo_r1 and an r_squared are not on a common scale and must not be compared.

Every requested line is computed on request, and the lines are also computed for a model that asked for none, so the table is available either way; drawn records whether the line appears on sca_plot().

Value

A data frame with one row per condition and reference line.

See Also

sca_analysis(), sca_table(), sca_terms()

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",
                   ceilings = "ce_fdh", scope = c(0, 100, 0, 100),
                   reference = c("ols", "qr"))
sca_reference(fit)
```

sca_relationship	<i>The direction stated as a claim about the relationship</i>
------------------	---

Description

Gives a direction in the wording of Table 1 of the framework. This is a claim about the theorised relationship, not about the shape of the estimated frontier, which may be a step function or a straight line under any of the four directions.

Usage

```
sca_relationship(direction, x = "X", y = "Y")
```

Arguments

- direction Character vector containing "HH", "LH", "HL" or "LL".
- x, y Names of the condition and the outcome.

Value

A character vector, one entry per direction.

See Also

[sca_empty_space\(\)](#), [sca_corner_map\(\)](#)

Examples

```
sca_relationship("HH")
sca_relationship("HH", "Training", "Performance")
```

sca_resample

How much the effect size moves when the sample does

Description

Refits one condition on random subsamples of several sizes and reports the distribution of the effect size at each size. Two things become visible: how far d moves under resampling at a given sample size, and how it trends as the sample grows.

Usage

```
sca_resample(
  model,
  x = NULL,
  ceiling = NULL,
  sizes = c(0.4, 0.6, 0.8),
  reps = 200,
  seed = NULL
)
```

Arguments

model	An object returned by sca_analysis() .
x	A condition name or position. Defaults to the first condition.
ceiling	Frontier technique. Defaults to the first estimated.
sizes	Subsample sizes. A value at or below 1 is a fraction of n , above 1 a count.
reps	Subsamples drawn per size.
seed	Optional seed. The caller's random stream is restored afterwards, as everywhere else in this package.

Value

A data frame with one row per subsample size.

Why there is no confidence interval

Deliberately, and for reasons specific to frontier estimators rather than general caution.

The ordinary n-out-of-n bootstrap is *inconsistent* here. Resampling with replacement duplicates observations, and a duplicated extreme point cannot push the envelope further out than the original did, so the resampled frontier lies inside the fitted one far more often than chance – the resampling distribution does not converge to the sampling distribution. This is the familiar result for DEA and FDH efficiency scores.

Subsampling without replacement, which is what this function does, restores consistency but leaves a bias with a known direction: a subsample of size $m < n$ has a *higher* expected effect size than the full sample, because the envelope is built from sample extremes and fewer points leave more of the corner empty by accident. A percentile interval read straight off the subsample distribution would therefore sit above the quantity it claims to cover. Correcting for that requires the estimator's convergence rate, which depends on the technique and the dimension and which this package does not assert.

So the table reports the distribution and stops there.

Reading it

`effect_size` is the full-sample value, repeated on every row as the anchor. `subsample_median` and the two percentiles describe the subsample distribution at that size.

The *spread* at one size answers "how much does my `d` depend on which cases I happened to collect". The *trend* down the `m` column answers "how much of my `d` is the sample being small" – it falls as `m` rises. Compare it with `effect_size_null` from `sca_table()`, which measures the same overstatement a different way, by random pairing at the full sample size.

Cost

One refit per replication per size. The default is 200 by 3, so a call is six hundred fits of one condition; nothing is cached between them.

See Also

`sca_scope_sensitivity()`, `sca_outliers()`, `sca_table()`

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv[seq(1, nrow(iv), by = 5), ], x = "X", y = "Y",
  direction = "HH", ceilings = "ce_fdh",
  scope = c(0, 100, 0, 100))
# A report would use the default reps = 200
sca_resample(fit, reps = 20, seed = 1)
```

sca_rescale

*Convert values to an SCA reporting scale***Description**

The conversion used internally by `sca_thresholds()`, exposed so that other packages can express their own quantities on exactly the same scale. This matters whenever two numbers are to be compared: a threshold produced elsewhere is only comparable with an SCA threshold if both were converted by the same rule, including the direction-aware mirroring and the percentile strictness that follows the inequality.

Usage

```
sca_rescale(
  values,
  reference,
  bounds = range(reference, na.rm = TRUE),
  scale = "percentage.range",
  letter = c("H", "L"),
  convention = c("absolute", "directional"),
  inequality = c("strict", "inclusive")
)
```

Arguments

values	Numeric vector in actual units.
reference	Observed values of the same variable, used by the percentile and sd scales.
bounds	Length-two axis bounds, including any theoretical scope. Defaults to the range of reference.
scale	One of the names returned by <code>sca_scales()</code> .
letter	"H" or "L", the direction letter for this axis.
convention	"absolute" or "directional".
inequality	"strict" or "inclusive"; affects the percentile scale only.

Value

A numeric vector on the requested scale. Non-finite inputs return NA.

Examples

```
x <- c(1, 3, 4, 7, 9)
sca_rescale(4, x, scale = "percentile", letter = "H")
sca_rescale(4, x, scale = "percentile", letter = "L")
```

sca_scales

Reporting scales available for SCA threshold tables

Description

Lists the scales and conventions understood by [sca_analysis\(\)](#) and [sca_thresholds\(\)](#), with a note on how each behaves when the sufficiency direction refers to a low level.

Usage

```
sca_scales()
```

Value

A data frame describing each scale.

Examples

```
sca_scales()
```

sca_scope

The scope each effect size was divided by

Description

Every effect size in this package is an empty area divided by the area of the X-Y scope, so the scope is half of every number reported. It is not always the scope that was asked for: the NCA engine widens a declared scope until it contains every observation, which changes the denominator without changing what the researcher wrote down.

Usage

```
sca_scope(model)
```

Arguments

model An object returned by [sca_analysis\(\)](#).

Details

This reports both. `declared_*` is NA when no scope was declared and the observed range was used. `used_*` is what the engine divided by, and `adjusted` is TRUE for a condition on which the two differ.

Value

A data frame with one row per condition.

See Also

[sca_table\(\)](#), [sca_analysis\(\)](#)

Examples

```
fit <- sca_analysis(
  subset(sca_four_configurations, configuration == "IV"),
  x = "X", y = "Y", direction = "HH", scope = c(0, 100, 0, 100)
)
sca_scope(fit)
```

`sca_scope_sensitivity` *How much of the effect size is the scope*

Description

Refits one condition on a series of scopes and reports the effect size each one produces. The scope is the denominator of every effect size and, unlike the frontier, it is chosen rather than estimated, so a reported d is a claim about the data and a claim about the box at the same time. This separates them.

Usage

```
sca_scope_sensitivity(
  model,
  x = NULL,
  ceiling = NULL,
  widen = c(0, 0.25, 0.5, 1),
  scopes = NULL
)
```

Arguments

<code>model</code>	An object returned by sca_analysis() .
<code>x</code>	A condition name or position. Defaults to the first condition.
<code>ceiling</code>	Frontier technique. Defaults to the first estimated.
<code>widen</code>	Fractions of the observed span to add to each side of each axis. 0 is the observed range.
<code>scopes</code>	Optional list of explicit <code>c(xmin, xmax, ymin, ymax)</code> boxes, used instead of <code>widen</code> when specific theoretical scopes are the point.

Details

By default the boxes are the observed range widened by a fraction of the observed span on each side, because that is a quantity a reader can judge: "a box half as wide again as the data" means something where a pair of numbers does not. The first row is always the observed range itself: the box the engine uses when no scope is declared, and the smallest box that contains the data, since any smaller one is widened back. When the model was fitted on a declared scope, that box is added as a final row labelled "as fitted (declared)", so the caller can see where their own choice sits among the alternatives instead of working it out by eye.

No column reports a change from a baseline, because which row is the baseline is the caller's question. The observed range answers "how much of this number is box, counting from the smallest box that holds the data"; the scope the model was fitted on answers "how sensitive is my number to the box I chose". Those are different questions and the table cannot silently pick one, so it reports the effect sizes and leaves the subtraction to the reader. The data does not change between rows: every difference in effect_size down the column is the box.

Value

A data frame with one row per scope.

See Also

[sca_scope\(\)](#), [sca_table\(\)](#)

Examples

```
fit <- sca_analysis(
  subset(sca_four_configurations, configuration == "IV"),
  x = "X", y = "Y", direction = "HH", ceilings = "ce_fdh"
)
sca_scope_sensitivity(fit)
```

sca_table	<i>Tidy SCA results</i>
-----------	-------------------------

Description

One row per condition and frontier technique.

Usage

```
sca_table(model, legacy = FALSE, p.adjust = "none")
```

Arguments

model	An object returned by <code>sca_analysis()</code> .
legacy	Append the column names retired in 0.4.0 and 0.5.0 as duplicates. See <code>sca_legacy_names()</code> .
p.adjust	Multiple-testing correction applied across the conditions within each frontier technique, as a method name understood by <code>stats::p.adjust()</code> . "none", the default, leaves the table exactly as it was. Any other method adds <code>p_value_adjusted</code> and <code>p_adjust_method</code> and re-derives significant, supported and support_status from the adjusted values.

Details

Three columns state the geometry in the vocabulary of the condition analysis in degree framework. `relationship` gives the direction as a claim about the theorised X-Y relationship, as in Table 1 of that framework. `empty_space` names what the expected empty space would contain. `boundary_type` says whether the boundary separating it is a ceiling, which limits how high Y can be for a given X, or a floor, which limits how low. For sufficiency the boundary is a floor whenever the outcome level is high and a ceiling whenever it is low, so the technique names CE-FDH and CR-FDH keep the word "ceiling" even where the fitted boundary is a floor.

`scope_area` is the denominator: `effect_size` is `empty_space_area / scope_area` exactly. `scope_adjusted` is TRUE where the engine had to widen the declared scope to hold every observation, so that the denominator is not the scope as declared. `sca_scope()` reports both scopes bound by bound.

`frontier` names the frontier technique the row was fitted with, such as "ce_fdh"; it is the value given in ceilings.

`empty_zone_area` is an exact copy of `empty_space_area` under the name it had before 0.5.0. It is kept because NSCA 0.4.5 reads it, and will be removed once a release of NSCA that reads `empty_space_area` has replaced that version. New code should use `empty_space_area`.

Value

A data frame with one row for every condition-frontier combination.

Multiple testing

Each condition carries its own permutation test, so a model with k conditions produces k p-values and a model with several frontier techniques produces k of them per technique. None of these is adjusted for multiplicity, and the Settings block printed by `print()` says so with the count and the arithmetic: were k tests independent, the chance of at least one false positive at alpha is $1 - (1 - \alpha)^k$.

Disclosure rather than adjustment, deliberately. Which correction fits depends on what the family is – whether the conditions were pre-specified, whether one significant condition would be reported as the finding or all of them together, whether the analysis is confirmatory at all. Those are properties of the design, not of the data, and a package that guessed at them would be making the methodological claim on the researcher's behalf. Adjust with `stats::p.adjust()` on the `p_value` column when the family is known.

The frontier techniques are a different problem and not a family-wise rate: two techniques are two estimates of one claim. Choosing between them after seeing both p-values is a garden of forking paths, which the Settings block also names.

See Also

[sca_thresholds\(\)](#), [sca_scope\(\)](#), [sca_corner_map\(\)](#), [sca_legacy_names\(\)](#)

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",
                  scope = c(0, 100, 0, 100))
sca_table(fit)

# Column names retired in earlier releases, appended as duplicates
names(sca_table(fit, legacy = TRUE))

# With a permutation test, on every fifth case to keep the example quick.
# A report would use 1000 or more permutations.
small <- iv[seq(1, nrow(iv), by = 5), ]
fit_test <- sca_analysis(small, x = "X", y = "Y", direction = "HH",
                      ceilings = "ce_fdh", scope = c(0, 100, 0, 100),
                      test.rep = 99, test.seed = 1)
sca_table(fit_test)
```

sca_terms

Terminology of condition analysis in degree

Description

A machine-readable glossary linking the vocabulary of the condition analysis in degree framework to the names this package uses. Terms retired in 0.4.0 are listed with their replacements in [sca_legacy_names\(\)](#).

Usage

```
sca_terms()
```

Value

A data frame with one row per term.

References

Chan, Y. (in press). *From Necessity to Sufficiency and Necessity-and-Sufficiency: Extending Condition Analysis in Degree*. University of Leeds.

See Also

[sca_legacy_names\(\)](#), [sca_corner_map\(\)](#), [sca_scales\(\)](#)

Examples

```
sca_terms()
```

sca_test_plot	<i>Plot an SCA permutation test</i>
---------------	-------------------------------------

Description

Plot an SCA permutation test

Usage

```
sca_test_plot(model, x = NULL, ceiling = NULL, bins = 30)
```

Arguments

- model An object returned by `sca_analysis()` with `test.rep > 0`.
- x A condition name or position. Defaults to the first condition.
- ceiling Frontier method.
- bins Number of histogram bins.

Value

A ggplot object.

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
# Every fifth case, to keep the permutation test quick
small <- iv[seq(1, nrow(iv), by = 5), ]
fit <- sca_analysis(small, x = "X", y = "Y", direction = "HH",
                    ceilings = "ce_fdh", scope = c(0, 100, 0, 100),
                    test.rep = 99, test.seed = 1)
sca_test_plot(fit)
```

sca_thresholds	<i>Sufficiency threshold table</i>
----------------	------------------------------------

Description

Turns the fitted frontier into directional sufficient-condition rules. A strict HH row reads $X > \text{threshold} \Rightarrow Y > \text{outcome level}$

Usage

```
sca_thresholds(
  model,
  ceiling = NULL,
  x = NULL,
  scale = NULL,
  outcome_scale = NULL,
  convention = NULL,
  inequality = c("strict", "inclusive"),
  digits = 6L,
  legacy = FALSE
)
```

Arguments

<code>model</code>	An object returned by sca_analysis() .
<code>ceiling</code>	A single frontier method that produces a threshold table.
<code>x</code>	Optional condition names or positions.
<code>scale, outcome_scale</code>	Reporting scales for the condition and outcome axes. Both default to the scales requested in sca_analysis() .
<code>convention</code>	"absolute" or "directional". Defaults to the convention requested in sca_analysis() .
<code>inequality</code>	Whether numeric rules use logically exact strict inequalities ("strict", the default) or an inclusive reporting convention ("inclusive").
<code>digits</code>	Significant digits used in the formatted rule text.
<code>legacy</code>	Append the column names retired in 0.4.0 and 0.5.0 as duplicates. See sca_legacy_names() .

Details

A *sufficiency threshold* is defined for one outcome target: in a higher X for higher Y relationship it is the lowest X value at which the estimated sufficiency frontier reaches that target. The table is the sufficiency counterpart of an NCA bottleneck table; the format is parallel but the interpretation is not, and the term "bottleneck" is reserved for necessity.

Values are stored in actual units and converted here, so a table can be re-expressed on any scale without refitting the model. Under the "absolute" convention 0 sits at the low end of each axis and the inequality carries the direction; under "directional" the axis is mirrored for a low-level direction, which flips the inequality so that every rule reads as "more of the sufficient thing". See [sca_scales\(\)](#).

The status column reports each row in sufficiency terms: "estimable" for a threshold inside the scope, "no_threshold" when no attainable condition value reaches the outcome level, and "always_satisfied" when every observation in scope already lies on the required side.

The inequalities describe an empirical empty-space pattern and should not be read as causal guarantees without an appropriate research design.

Value

A long-form data frame of threshold rules.

See Also

[sca_table\(\)](#), [sca_legacy_names\(\)](#), [sca_scales\(\)](#)

Examples

```
iv <- subset(sca_four_configurations, configuration == "IV")
fit <- sca_analysis(iv, x = "X", y = "Y", direction = "HH",
                   ceilings = "ce_fdh", scope = c(0, 100, 0, 100))
sca_thresholds(fit)
sca_thresholds(fit, scale = "actual", inequality = "inclusive")
```

Index

- * **datasets**
 - sca_four_configurations, 17
- NCA::nca_analysis(), 5–7
- NSCA::nsca(), 6
- NSCA::nsca_analysis(), 8
- print(), 37
- sca, 5
- sca(), 4
- sca_analysis, 6
- sca_analysis(), 4–6, 11, 12, 16, 18, 20, 23–26, 29–31, 34, 35, 37, 39, 40
- sca_boundary_type, 9
- sca_boundary_type(), 13
- sca_check_direction, 10
- sca_compare, 10
- sca_compare(), 3, 4, 18, 20–22, 24
- sca_corner, 12
- sca_corner(), 10
- sca_corner_location, 13
- sca_corner_location(), 9
- sca_corner_map, 14
- sca_corner_map(), 10, 13, 16, 31, 38
- sca_direction, 15
- sca_empty_space, 15
- sca_empty_space(), 31
- sca_extract, 16
- sca_extract(), 19
- sca_four_configurations, 4, 17
- sca_legacy_names, 18
- sca_legacy_names(), 16, 37, 38, 40, 41
- sca_locationshift, 19
- sca_locationshift(), 4, 12, 18
- sca_magnitude, 20
- sca_normalize, 22
- sca_outliers, 23
- sca_outliers(), 4, 32
- sca_output, 24
- sca_output(), 4
- sca_plot, 25
- sca_plot(), 4, 30
- sca_power, 26
- sca_power(), 3, 28
- sca_powerplot, 27
- sca_random, 28
- sca_reference, 29
- sca_reference(), 5, 7, 12
- sca_relationship, 30
- sca_relationship(), 16
- sca_resample, 31
- sca_resample(), 4
- sca_rescale, 33
- sca_scales, 34
- sca_scales(), 8, 33, 38, 40, 41
- sca_scope, 34
- sca_scope(), 4, 18, 36–38
- sca_scope_sensitivity, 35
- sca_scope_sensitivity(), 4, 32
- sca_table, 36
- sca_table(), 3, 4, 12, 19, 21, 22, 29, 30, 32, 35, 36, 41
- sca_terms, 38
- sca_terms(), 9, 19, 30
- sca_test_plot, 39
- sca_test_plot(), 4
- sca_thresholds, 39
- sca_thresholds(), 3, 4, 8, 19, 25, 33, 34, 38
- SCAtools (SCAtools-package), 3
- SCAtools-package, 3
- stats::p.adjust(), 37