

Package ‘NSCA’

October 10, 2026

Type Package

Title Necessary and Sufficient Condition Analysis

Version 0.4.5

Date 2026-09-30

Description Tests and quantifies bivariate statements in which a condition is both necessary and sufficient for an outcome. Necessity and sufficiency are estimated as two empty-space frontiers on diagonally opposite corners of the same scatter plot, using the 'NCA' package of Dul (2016) [doi:10.1177/1094428115584005](https://doi.org/10.1177/1094428115584005) for the necessity side and 'SCAtools' for the sufficiency side. The two empty zones and the region between them tile the analytic scope, which yields the reported areas as one identity rather than as separate definitions: a weakest-component effect, a normalised geometric mean of the two components, the share of the scope the claims jointly rule out, and the data zone they still admit. The conjunction is decided by an intersection-union combination of the two directional permutation tests, optionally driven by a single shared permutation sequence so that a statistic of both components can also be tested. These are random-pairing screens rather than direct tests of a causal necessary-and-sufficient relation, no magnitude benchmarks are asserted for the joint indices, and a dual-threshold table separates outcome levels that are out of reach, admitted, or guaranteed. An ordinary least-squares line can be drawn alongside the two frontiers as a central-tendency reference; it is an average-effect summary and is never treated as a component of either claim.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports ggplot2 (>= 3.4.0), NCA (>= 5.0.2), SCAtools (>= 0.4.1), stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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

BugReports <https://github.com/youngchanresearcher/NSCA/issues>
RoxygenNote 7.3.2
NeedsCompilation no
Author Young Chan [aut, cre]
Maintainer Young Chan <youngchanresearcher@gmail.com>
Repository CRAN
Date/Publication 2026-10-10 11:00:08 UTC

Contents

nsca	2
nsca_analysis	3
nsca_corners	6
nsca_direction_map	7
nsca_extract	8
nsca_joint	9
nsca_legacy_names	10
nsca_plot	11
nsca_reference	12
nsca_results	13
nsca_table	14
nsca_terms	17
nsca_thresholds	18
Index	21

nsca	<i>Quick necessary and sufficient condition analysis</i>
------	--

Description

A thin wrapper over `nsca_analysis()` with a permutation test attached.

Usage

```
nsca(  
  data,  
  x,  
  y,  
  direction = "HH",  
  ceilings = "ce_fdh",  
  reference = NULL,  
  test.rep = 1000  
)
```

Arguments

<code>data</code>	A data frame or object coercible to a data frame.
<code>x</code>	Columns containing one or more conditions.
<code>y</code>	A single outcome column.
<code>direction</code>	Necessary-and-sufficient direction(s): "HH", "LH", "HL", or "LL". The first letter is the condition level and the second the outcome level.
<code>ceilings</code>	One or more empty-space frontier techniques, applied identically to both sides. "ols" is rejected here because it estimates central tendency rather than an empty space; pass it as <code>reference</code> instead.
<code>reference</code>	Central-tendency lines drawn beside the two frontiers 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 is an average-effect summary, enters no joint index, no threshold and no support decision, and is reported by nsca_reference() .
<code>test.rep</code>	Number of permutation resamples for the engines' own component tests. Zero skips testing, and without it no joint-support verdict can be reached.

Value

An object of class `nsca_result`.

See Also

[nsca_analysis\(\)](#)

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))

# The default 1000 permutations take several seconds.
nsca(dat, "X", "Y")
```

nsca_analysis

Necessary and sufficient condition analysis

Description

Estimates the two empty-space frontiers implied by a necessary-and-sufficient statement and assesses them jointly. Sufficiency is delegated to [SCAtools::sca_analysis\(\)](#) and necessity to [NCA::nca_analysis\(\)](#), on diagonally opposite corners of the same scatter plot.

Usage

```

nsca_analysis(
  data,
  x,
  y,
  direction = "HH",
  ceilings = c("ce_fdh", "cr_fdh"),
  reference = 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,
  test.rep = 0,
  test.p_confidence = 0.95,
  test.p_threshold = 0.05,
  relevance = NULL,
  shared.test.rep = 0,
  shared.seed = NULL,
  geometry.max_overlap = 0.01,
  geometry.max_reconstruction = 0.02,
  purity = FALSE
)

```

Arguments

<code>data</code>	A data frame or object coercible to a data frame.
<code>x</code>	Columns containing one or more conditions.
<code>y</code>	A single outcome column.
<code>direction</code>	Necessary-and-sufficient direction(s): "HH", "LH", "HL", or "LL". The first letter is the condition level and the second the outcome level.
<code>ceilings</code>	One or more empty-space frontier techniques, applied identically to both sides. "ols" is rejected here because it estimates central tendency rather than an empty space; pass it as reference instead.
<code>reference</code>	Central-tendency lines drawn beside the two frontiers 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 is an average-effect summary, enters no joint index, no threshold and no support decision, and is reported by nsca_reference() .
<code>scope</code>	Optional theoretical scope <code>c(xmin, xmax, ymin, ymax)</code> , shared by both sides.
<code>threshold.x, threshold.y</code>	Reporting scales for nsca_thresholds() , as in SCAtools::sca_analysis() .
<code>convention</code>	Reporting convention, as in SCAtools::sca_analysis() .

<code>steps</code>	Number of outcome levels, or an explicit vector of levels on the threshold.y scale.
<code>step.size</code>	Optional spacing between outcome levels.
<code>cutoff</code>	How out-of-range threshold values are represented.
<code>qr.tau</code>	Quantile used by the quantile-regression frontier.
<code>test.rep</code>	Number of permutation resamples for the engines' own component tests. Zero skips testing, and without it no joint-support verdict can be reached.
<code>test.p_confidence</code>	Confidence level for permutation p-value accuracy.
<code>test.p_threshold</code>	Significance level used by the intersection-union combination of the two directional component tests.
<code>relevance</code>	Optional pre-specified practical-relevance thresholds for the component effect sizes, as one number applied to both or <code>c(necessity, sufficiency)</code> . 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.
<code>shared.test.rep</code>	Number of replications of the shared permutation sequence. Zero, the default, skips it; <code>p_weakest_perm</code> is then NA and the component p-values come from the engines.
<code>shared.seed</code>	Optional seed for the shared sequence. The caller's random stream is restored afterwards.
<code>geometry.max_overlap</code>	Largest excess frontier overlap, beyond what a step frontier produces by construction, that still counts as acceptable geometry.
<code>geometry.max_reconstruction</code>	Largest residual of the area identity that still counts as acceptable geometry.
<code>purity</code>	Compute the engine's extra purity metrics where it offers them. Off by default: the engine computes them only for a corner with neither axis flipped, so exactly one side of an NSCA model can ever have them, and which side depends on the direction. No NSCA statistic uses them, and they are expensive on data with many frontier points.

Value

An object of class `nsca_result`.

Matched estimation

Both sides are always estimated with the same frontier technique, the same theoretical scope, the same observations and the same outcome levels. Every joint index compares the two empty zones, and a comparison is only meaningful when both sides are measured the same way: envelopment frontiers hug the data and yield the largest empty areas, while regression frontiers cut into them, so mixing techniques across sides would decide which side looks weaker by the choice of technique rather than by the evidence. Supplying several techniques produces one internally matched row per technique, which is the right way to check whether a conclusion survives that choice.

Degenerate scopes

A constant outcome is refused outright when no theoretical scope is given, because it has no scope of its own and every empty area would be measured against a scope of zero height. With a scope imposed the analysis runs but warns, and marks the affected rows degenerate in `nsca_table()`. An axis that does not move inside the declared scope makes every reported area a property of the scope rather than of the data, and it does so flatteringly: a constant outcome at the centre of the scope leaves the upper and lower halves both empty, so both components reach 0.5, the admissible region closes, and every joint index reports perfect joint support for data carrying no information. A milder warning fires when the observations span less than five per cent of the declared scope on either axis.

The shared permutation sequence

`shared.test.rep` replaces the engines' two separate permutation runs with one sequence that drives both. Each replication shuffles the outcome once, refits both engines on that same shuffled frame, and records `d_nec`, `d_suf` and their minimum together. This is what makes `p_weakest_perm` meaningful: the null distribution of a statistic of both components depends on how they co-vary under random pairing, and independent permutations would destroy exactly that dependence. When it is used, `p_nec` and `p_suf` are taken from the same sequence, the `p_source` column reports "shared", and all four p-values are mutually consistent. It costs two engine fits per replication, so it is off by default.

See Also

`nsca_table()`, `nsca_results()`, `nsca_joint()`, `nsca_thresholds()`, `nsca_corners()`

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))
fit <- nsca_analysis(dat, "X", "Y", direction = "HH", ceilings = "ce_fdh")
fit
```

nsca_corners

Empty corners of a necessary-and-sufficient statement

Description

Returns the two physically empty scatter-plot corners implied by a necessary-and-sufficient statement. The necessity corner is always the diagonal opposite of the sufficiency corner.

Usage

```
nsca_corners(direction = "HH")
```

Arguments

`direction` One or more of "HH", "LH", "HL", "LL". The first letter refers to the level of the condition and the second to the level of the outcome.

Value

A data frame with the necessity and sufficiency corners and the side of the point cloud each frontier bounds.

Examples

```
nsca_corners("HH")
nsca_corners(c("HH", "LL"))
```

<code>nsca_direction_map</code>	<i>Complete mapping between directions, corners and component state-ments</i>
---------------------------------	---

Description

Reproduces Table 1 of the condition analysis in degree framework, which sets out the main directional types of the X-Y relationship together with the expected empty space and boundary each implies for necessity and for sufficiency, and adds the corner numbering the frontier engines use.

Usage

```
nsca_direction_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. Theory may instead posit a curvilinear or U-shaped relationship, for which the expected empty space is no longer a single corner; those are outside the scope of this package.

Value

A data frame describing all four necessary-and-sufficient directions. The biconditional column keeps its name: it is the joint statement written out, and the short logical word is the more usable column name.

See Also

[nsca_corners\(\)](#), [nsca_terms\(\)](#)

Examples

```
nsca_direction_map()
```

nsca_extract	<i>Extract one value from an NSCA result</i>
--------------	--

Description

Extract one value from an NSCA result

Usage

```
nsca_extract(model, x = NULL, ceiling = NULL, param = "weakest_effect")
```

Arguments

model	An object returned by nsca_analysis() .
x	Condition name. Defaults to the first condition.
ceiling	Frontier technique. Defaults to the first requested.
param	A column of nsca_table() , one of the retired names in nsca_legacy_names() , or a parameter name understood by the underlying engines, prefixed with "nec:" or "suf:".

Value

The selected value.

See Also

[nsca_table\(\)](#), [nsca_legacy_names\(\)](#)

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))
fit <- nsca_analysis(dat, "X", "Y", ceilings = "ce_fdh")
nsca_extract(fit, param = "weakest_effect")
nsca_extract(fit, param = "nec:Ceiling accuracy")
```

nsca_joint	<i>Joint strength of the two components</i>
------------	---

Description

Combines a necessity effect size and a sufficiency effect size into one joint index, with an explicit choice of how much the stronger component may compensate for the weaker one.

Usage

```
nsca_joint(d_nec, d_suf, p = 0, normalize = TRUE)
```

Arguments

d_nec, d_suf	Necessity and sufficiency effect sizes. Vectors are recycled to a common length.
p	Degree of the power mean; the degree of compensation. Defaults to 0, the geometric mean.
normalize	Multiply by two so that the index spans [0, 1]. The guarantee rests on $M_p \leq (d_{nec} + d_{suf})/2 \leq 0$. and therefore holds only for $p \leq 1$.

Details

The index is the power mean $M_p = ((d_{nec}^p + d_{suf}^p)/2)^{1/p}$, optionally doubled so that it spans [0, 1]. The parameter p is the degree of compensation:

$p = -\text{Inf}$ min(d_nec, d_suf). Fully non-compensatory: no amount of strength on one side raises the index if the other side is weak. This is the `weakest_effect` column of `nsca_table()`, reported there unnormalised so that it stays on the components' own scale.

$p = -1$ The harmonic mean. Less compensatory than the geometric mean, still zero as soon as either component is zero.

$p = 0$ The geometric mean. **Partially** compensatory: a larger component does raise the index, but at a diminishing rate, and the index still collapses to zero if either component does. This is the `balanced_joint_effect` column of `nsca_table()`.

$p = 1$ The arithmetic mean, that is, half the sum. Fully compensatory: one strong component alone can carry the index, so a necessary-only relation scores as highly as a necessary-and-sufficient one. This is why the sum is a poor conjunction summary, but it is the same family, not a different kind of quantity.

The geometric mean is the middle course, and describing it as a non-compensatory "AND" would overstate it. What it does is penalise asymmetry: at an equal sum of 0.80, (0.40, 0.40) gives 0.80 and (0.70, 0.10) gives 0.529.

No magnitude benchmarks are supplied. Conventions for a single NCA effect size do not transfer, because this index has a different scale and a different null: under independence both components carry a positive finite-sample bias that a product of the two amplifies rather than cancels, and its size depends on n, the frontier technique and the scope. Calibrate by simulation on the design at hand before attaching words to values.

Value

A numeric vector, NA where either component is missing or negative.

See Also

[nsca_table\(\)](#), [nsca_analysis\(\)](#)

Examples

```
nsca_joint(0.40, 0.40)           # balanced
nsca_joint(0.70, 0.10)           # same sum, penalised for asymmetry
nsca_joint(0.40, 0.40, p = -Inf) # the minimum, normalised
nsca_joint(0.40, 0.40, p = -Inf, normalize = FALSE) # weakest_effect
```

nsca_legacy_names	<i>Names retired in NSCA 0.3.0 and 0.4.0</i>
-------------------	--

Description

The 0.3.0 vocabulary followed the geometry rather than the interpretation: the region between the two frontiers was named for what it is and the two joint indices for what they do to a weak component rather than for a verdict.

Usage

```
nsca_legacy_names()
```

Details

0.4.0 goes further and adopts the vocabulary of *condition analysis in degree*. The region compatible with both components is the **admissible region**; the distance on the X scale between the two thresholds at one outcome target is the **necessity-sufficiency interval**; evidence for both components is **joint support**; and *boundary* is reserved for the theoretical line an expected empty space is separated by, so the argument controlling the strictness of a reported inequality became *inequality*.

Old names remain usable. [nsca_table\(\)](#) and [nsca_thresholds\(\)](#) append them as duplicate columns when called with `legacy = TRUE`, [nsca_extract\(\)](#) accepts them with a warning, and the retired arguments warn but still work.

Value

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

See Also

[nsca_terms\(\)](#)

Examples

```
nsca_legacy_names()
```

nsca_plot

*Plot both frontiers***Description**

Draws the scatter plot with the necessity and sufficiency frontiers and shades the admissible region between them. Several frontier techniques may be drawn at once; unlike the computation, a chart compares nothing numerically, so mixing techniques here is safe and often informative.

Usage

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

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

Arguments

<code>model</code>	An object returned by <code>nsca_analysis()</code> .
<code>x</code>	Optional condition names or positions.
<code>ceilings</code>	Frontier techniques to draw. Defaults to all that were estimated.
<code>shade</code>	Shade the admissible region under the first technique.
<code>annotate</code>	Label the two corners that the claim requires to be empty.
<code>...</code>	Passed on to <code>nsca_plot()</code> by the <code>plot()</code> method.

Value

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

See Also

`nsca_analysis()`, `nsca_table()`

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))
fit <- nsca_analysis(dat, "X", "Y", ceilings = "ce_fdh")
nsca_plot(fit)
plot(fit)
```

nsca_reference	<i>Central-tendency reference line</i>
----------------	--

Description

Fits the ordinary least-squares regression of the outcome on each condition, on the same observations both frontiers were fitted to. This is the line `lm(y ~ x)` returns, and it is what `reference = "ols"` draws on `nsca_plot()`.

Usage

```
nsca_reference(model, x = NULL)
```

Arguments

<code>model</code>	An object returned by <code>nsca_analysis()</code> .
<code>x</code>	Optional condition names or positions. All conditions by default.

Details

It is reported apart from `nsca_table()` because it is a different kind of quantity. Regression describes how the expected outcome moves with the condition; the two frontiers describe which condition-outcome combinations are absent. Section 6.1 of the condition analysis in degree framework keeps the two logics side by side rather than merging them 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 zone, no effect size, no p value, no threshold, and no part in the joint-support decision.

The line is computed on request, so it is available whether or not `reference = "ols"` was set; drawn records whether it also appears on the plot.

Value

A data frame with one row per condition: `condition`, `outcome`, `line`, `line_type`, `observations`, `intercept`, `slope`, `r_squared` and `drawn`.

See Also

`nsca_analysis()`, `nsca_table()`, `nsca_terms()`

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))
fit <- nsca_analysis(dat, "X", "Y", ceilings = "ce_fdh", reference = "ols")
nsca_reference(fit)
```

nsca_results	<i>All three NSCA result layers</i>
--------------	-------------------------------------

Description

Returns the necessity analysis, the sufficiency analysis, and their joint necessary-and-sufficient analysis as three explicit tables. The third does not represent an independent third experiment: its `p_nsca_iut` is the conjunction $\max(p_nec, p_suf)$ and can pass only when both component screens pass.

Usage

```
nsca_results(model)

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

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

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

Arguments

<code>model</code>	An object returned by <code>nsca_analysis()</code> .
<code>x</code>	An object returned by <code>nsca_analysis()</code> , for the print method.
<code>...</code>	Additional arguments reserved for methods.
<code>object</code>	An object returned by <code>nsca_analysis()</code> .

Value

A named list containing necessity, sufficiency, and necessary_and_sufficient data frames. The methods `print`, or return a summary object.

See Also

`nsca_table()`

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))
fit <- nsca_analysis(dat, "X", "Y", ceilings = "ce_fdh")
layers <- nsca_results(fit)
names(layers)
```

```
fit
summary(fit)
```

nsca_table	<i>Joint necessity and sufficiency results</i>
------------	--

Description

One row per condition and frontier technique. Both sides of a row always use the same technique, the same observations and the same scope.

Usage

```
nsca_table(model, legacy = FALSE)
```

Arguments

model	An object returned by nsca_analysis() .
legacy	Append the column names retired in 0.3.0 and 0.4.0 as duplicates. See nsca_legacy_names() .

Value

A data frame of joint results.

What is primary

The component effect sizes `d_nec` and `d_suf` are primary. Everything else is a way of putting them together, and three such ways are reported rather than one, because nothing in the geometry of two empty zones selects one.

`weakest_effect` $\min(d_{nec}, d_{suf})$, the fully non-compensatory summary. It falls to zero as soon as either component does and stays on the components' own scale, so its maximum is about 0.5 rather than 1.

`balanced_joint_effect` $2 * \sqrt{d_{nec} * d_{suf}}$, normalised onto $[0, 1]$. Partially compensatory: a larger component raises it at a diminishing rate, and it still collapses to zero if either component does. At an equal component sum of 0.80, $(0.40, 0.40)$ gives 0.80 and $(0.70, 0.10)$ gives 0.529. It is *not* a measure of balance: $(0.50, 0.125)$ and $(0.25, 0.25)$ both score 0.50. The balance column measures that separately.

`joint_empty_zone_coverage` The share of the scope that at least one claim rules out, $1 - \text{admissible_region_share}$.
 Curvature-neutral: it reaches 1 whenever the admissible region closes, whatever the shape of the relation.

`min` and the geometric mean are the power means at $p = -\text{Inf}$ and $p = 0$, so choosing between them is choosing how much compensation to allow; the component sum sits at $p = 1$, the fully compensatory end, and is a poor conjunction summary because one strong component can carry it. [nsca_joint\(\)](#) exposes the family directly.

The area identity

The two empty spaces and the admissible region tile the scope, so by inclusion and exclusion

$$\text{admissible_region_share} = 1 - d_{nec} - d_{suf} + O$$

where O is overlap_share. Everything else follows from it rather than being asserted separately. Because admissible_region_share is a share of the scope it lies in $[0, 1]$ unconditionally, and therefore so does joint_empty_zone_coverage. The other two indices are bounded only up to the overlap:

$$d_{nec} + d_{suf} \leq 1 + O, \quad \text{weakest_effect} \leq 0.5 + O/2, \quad \text{balanced_joint_effect} \leq 1 + O$$

Both bounds are tight and are attained exactly by a perfect diagonal, where O is $1 / (n - 1)$: at $n = 40$ the observed values are 0.5128 and 1.0256 against bounds of 0.5128 and 1.0256. The excess is discretization, not evidence, and the indices are left unclamped so that it stays visible.

Vocabulary

The column names follow the condition analysis in degree framework. relationship states the direction as a claim about the theorised X-Y relationship, as in Table 1 of that framework, not about the shape of the estimated frontier. necessity_empty_space and sufficiency_empty_space name what each expected empty space would contain, and necessity_boundary and sufficiency_boundary say whether the boundary separating it is a ceiling, which limits how high the outcome can be for a given condition value, or a floor, which limits how low. The two are always one of each, because the two corners are diagonally opposite.

The disjointness check

Whenever the two frontiers do not cross, the floor lies at or below the ceiling throughout the scope, so no point belongs to both expected empty spaces. Because each component effect size is that space's share of the same scope, the two shares cannot sum to more than one. Component effect sizes summing above one therefore say the frontiers have crossed and the joint result needs diagnosis before interpretation, rather than that it is unusually strong. component_sum reports $d_{nec} + d_{suf}$ so that this rule can be applied directly.

frontiers_crossed answers the same question from the measurement rather than from the sum. The sum is a proxy for a crossing that is not otherwise available; here it is available, because overlap_share is the area on which the two empty spaces actually overlap, integrated from the reconstructed frontiers. Using the measurement avoids the proxy's failure mode, which is that the sum and the allowance below are reached by different routes and can disagree in the last few digits on exactly the configuration the check is meant to bless.

That allowance exists because a step frontier overlaps itself by construction. Between two consecutive observations the upper staircase still holds the earlier value while the lower one has already moved, so the two spaces overlap on every tread; summed over the $n - 1$ treads that is exactly $1 / (n - 1)$ of the scope, attained by a perfect diagonal, which is the cleanest relation there is rather than a failure. For every smooth frontier the allowance is zero and any overlap at all is a genuine crossing.

`frontiers_crossed` asks whether the frontiers crossed at all; `geometry_acceptable` asks whether they crossed by more than `geometry.max_overlap`. A row can be flagged as crossed and still have acceptable geometry.

Geometry diagnostics

`reconstruction_error` is the residual of the identity above. The two sides come from different places: `admissible_region_share` and `overlap_share` are integrated from the frontiers this package reconstructs, while `d_nec` and `d_suf` are reported by the engines from their own internal frontiers. A residual near zero says the two agree; a large one says the reconstruction has drifted from what the engine actually fitted, which is the one failure mode that a plot cannot reveal, because a wrongly reconstructed frontier still looks like a frontier.

`degenerate` marks a condition whose axis does not vary inside the declared scope. `geometry_acceptable` combines both diagnostics with the excess frontier overlap, allowing a step frontier the $1 / (n - 1)$ it produces by construction.

Decisions

`p_nsca_iut` is $\max(p_{nec}, p_{suf})$, the intersection-union combination. It passes only when both components reject at `test.p_threshold`, and it is level-alpha without any assumption that the two component tests are independent, so no correction is needed merely for combining two pre-specified tests. Multiplicity across several conditions or frontiers is a separate matter. The combination is also typically conservative, so an NSCA design needs more observations than either component alone.

`p_weakest_perm` tests $\min(d_{nec}, d_{suf})$ directly and is available only when `shared.test.rep` was used, because the null distribution of a statistic of both components depends on how they co-vary under random pairing. It answers a different question from the intersection-union rule and does not replace it: its null is random pairing alone, which is a proper subset of the union null "not necessary or not sufficient", so it carries no level-alpha guarantee over that union. Treat it as a sensitivity analysis. `p_source` reports whether the component p-values came from the engines' own separate runs or from the shared sequence.

`necessity_supported` and `sufficiency_supported` require significance and, when a relevance threshold was pre-specified, an effect at least that large. `joint_support` additionally requires `geometry_acceptable`, so a degenerate or badly reconstructed row cannot be reported as supported.

Blind spot

All three joint summaries are areas over the declared scope and share one blind spot. A constant outcome at the centre of an imposed scope leaves the upper and lower halves both empty, so both components reach 0.5, the data zone closes, and every index reports perfect joint support for data carrying no information. `balanced_joint_effect` is the most exposed, since `weakest_effect` at least follows the weaker side once the constant sits off centre. The degenerate flag and the permutation screen are what cover it: permuting a constant outcome reproduces the same geometry every time, so every p-value goes to 1.

No magnitude benchmarks are supplied for any index. Conventions for a single NCA effect size do not transfer: the scales differ, and under independence both components carry a positive finite-sample bias that `balanced_joint_effect` amplifies rather than cancels, by an amount depending

on the sample size, the frontier technique and the scope. Calibrate by simulation on the design at hand.

See Also

[nsca_results\(\)](#), [nsca_joint\(\)](#), [nsca_extract\(\)](#), [nsca_legacy_names\(\)](#)

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))
fit <- nsca_analysis(dat, "X", "Y", ceilings = "ce_fdh")
nsca_table(fit)
```

nsca_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. Names retired along the way are listed by [nsca_legacy_names\(\)](#).

Usage

```
nsca_terms()
```

Value

A data frame with one row per term.

See Also

[nsca_legacy_names\(\)](#), [nsca_direction_map\(\)](#), [nsca_table\(\)](#)

Examples

```
nsca_terms()
```

nsca_thresholds	<i>Dual thresholds and the necessity-sufficiency interval</i>
-----------------	---

Description

For every outcome level, reports the necessity and sufficiency thresholds and the direction-oriented distance between them.

Usage

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

Arguments

model	An object returned by nsca_analysis() .
ceiling	A single frontier technique, applied to both sides.
x	Optional condition names or positions.
scale, outcome_scale	Reporting scales, defaulting to those requested in nsca_analysis() . See SCAtools::sca_scales() .
convention	"absolute" or "directional".
inequality	Strict or inclusive inequalities. Renamed from boundary in 0.4.0, which is reserved for the theoretical line an expected empty space is separated by.
digits	Significant digits in the formatted text.
legacy	Append the column names retired in 0.3.0 and 0.4.0 as duplicates. See nsca_legacy_names() .
boundary	Deprecated. Former name of inequality.

Details

For a fixed outcome target, the *necessity-sufficiency interval* is the distance on the condition axis between the two thresholds: if a target requires at least 40 units of the condition but 70 units are enough for it, the interval runs from 40 to 70. It is not a confidence interval: it carries no statistical uncertainty.

necessity_sufficiency_interval and overlap_width are reported on scale, the same scale as the two threshold columns they are the distance between, so a row can be checked by subtraction. necessity_sufficiency_interval_actual and overlap_width_actual keep the same two quantities in the units of the data whatever scale is asked for, and threshold_gap_actual is the signed actual-unit gap the pair is split from. Before 0.4.3 the two unsuffixed columns held actual units on every scale, which put three columns of two different kinds in one printed row.

Under scale = "percentile" the reported column is a difference of ranks rather than a distance, so it reads as the share of cases lying between the two thresholds. That is a useful quantity, but it is not a width: equal percentile gaps in a dense and in a sparse part of the distribution stand for very different distances on the condition axis. For a normalised width that is comparable across studies use "percentage.range" or "sd".

The interval is also distinct from the component effect sizes. Effect sizes aggregate exclusion over the whole scope, whereas the interval is read at one outcome target, so large component effects do not guarantee a narrow interval at any particular level and a narrow interval at one target says nothing about the others. Reporting both, at several targets, is not redundant.

Value

A data frame with one row per condition and outcome level.

The three regions

Necessity and sufficiency answer different questions about the same outcome level, and together they cut the condition axis into three parts. For a high-X direction:

below necessity_threshold The outcome level is out of reach.

between the two thresholds It is admitted but not guaranteed. This is the admissible interval, and necessity_sufficiency_interval is its width. It is the horizontal cross-section, at one outcome level, of the region whose area admissible_region_share reports.

above sufficiency_threshold The fitted frontier guarantees it.

For a low-X direction these inequalities reverse: values above the necessity threshold are out of reach and values below the sufficiency threshold are guaranteed. threshold_gap_actual is direction-oriented, so a positive value means an admitted interval in all four directions. A negative value is reported as "frontiers overlap" in threshold_status, with its magnitude in overlap_width and overlap_width_actual, rather than being silently truncated to zero: the two thresholds are then mutually incompatible at that level, which is a modelling problem and not a narrow admissible interval.

The interval collapses to zero exactly where the relation is deterministic at that level, reported as "exact correspondence": one X value is then both the minimum required and the sufficient level. An estimated interval of zero is *consistent* with exact correspondence but does not by itself establish it, and a small non-zero interval should be called near correspondence only when a substantively justified tolerance has been defined. A positive interval does not negate joint support; it indicates imperfect threshold correspondence. Neither NCA nor SCA alone can produce this table: each supplies one edge of the interval.

The two out-of-range markers read differently on the two sides, because their meaning inverts under contraposition. On the necessity side they keep their necessity reading, giving "no_minimum",

"no_maximum" and "unattainable"; on the sufficiency side SCAtools supplies the sufficiency reading.

See Also

[nsca_table\(\)](#), [nsca_analysis\(\)](#), [nsca_legacy_names\(\)](#)

Examples

```
set.seed(1)
x <- sort(runif(60))
dat <- data.frame(X = x, Y = pmin(pmax(x + rnorm(60, 0, 0.1), 0), 1))
fit <- nsca_analysis(dat, "X", "Y", ceilings = "ce_fdh")
nsca_thresholds(fit)
```

Index

NCA::nca_analysis(), 3
nscs, 2
nscs_analysis, 3
nscs_analysis(), 2, 3, 8, 10–14, 18, 20
nscs_corners, 6
nscs_corners(), 6, 7
nscs_direction_map, 7
nscs_direction_map(), 17
nscs_extract, 8
nscs_extract(), 10, 17
nscs_joint, 9
nscs_joint(), 6, 14, 17
nscs_legacy_names, 10
nscs_legacy_names(), 8, 14, 17, 18, 20
nscs_plot, 11
nscs_plot(), 12
nscs_reference, 12
nscs_reference(), 3, 4
nscs_results, 13
nscs_results(), 6, 17
nscs_table, 14
nscs_table(), 6, 8–13, 17, 20
nscs_terms, 17
nscs_terms(), 7, 10, 12
nscs_thresholds, 18
nscs_thresholds(), 4, 6, 10

plot.nscs_result (nscs_plot), 11
print.nscs_result (nscs_results), 13
print.summary_nscs_result
 (nscs_results), 13

SCAtools::sca_analysis(), 3, 4
SCAtools::sca_scales(), 18
summary.nscs_result (nscs_results), 13