Package 'NCA'

June 17, 2026

Type Package

Title Necessary Condition Analysis

Version 5.0.2

Date 2026-06-16

Description Performs a Necessary Condition Analysis (NCA). (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>.
NCA identifies necessary (but not sufficient) conditions in datasets, where x causes (e.g. precedes) y. Instead of drawing a regression line "through the middle of the data" in an xy-plot, NCA draws the ceiling line. The ceiling line y = f(x) separates the area with observations from the area without observations.
(Nearly) all observations are below the ceiling line: y <= f(x). The empty zone is in the upper left hand corner of the xy-plot (with the convention that the x-axis is "horizontal" and the y-axis is "vertical" and that values increase "upwards" and "to the right"). The ceiling line is a (piecewise) linear non-decreasing line: a linear step function or a straight line. It indicates which level of x (e.g., an effort, a characteristic) is necessary but not sufficient for a (desired or undesired) level of y (e.g., good performance or disease). A quick start guide for using this package can be found here: <https://repub.eur.nl/pub/78323/> or <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2624981>.

URL https://www.eur.nl/en/erim/erim/research-initiatives/necessary-condition-analysis

License GPL (>= 3)

Depends R (>= 3.5.0)

Imports gplots, quantreg, KernSmooth, lpSolve, ggplot2, doParallel, foreach, iterators, plotly, truncnorm, DBI, RSQLite

NeedsCompilation no

Repository CRAN

Suggests testthat

Author Jan Dul [aut], Govert Buijs [cre]

Maintainer Govert Buijs <buijs@rsm.nl>

Date/Publication 2026-06-17 14:50:11 UTC

Contents

`NCA-package` *Necessary Condition Analysis*

Description

The NCA package implements Necessary Condition Analysis (NCA) as developed by Dul (2016). For running the NCA package a data file (e.g., mydata.csv, which contains the input data) must be available. An example data file (presented in above article) is included in the package. The user must load the data and call the nca function.

Details

Package:	NCA
Type:	Package
Version:	5.0.2
Date:	2026-06-16
License:	GPL (>= 3)

Author(s)

Author: Jan Dul <jdul@rsm.nl>
Maintainer: Govert Buijs <buijs@rsm.nl>

References

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
Dul, J. (2020). Conducting Necessary Condition Analysis. Sage publishers. ISBN: 9781526460141. https://uk.sagepub.com/en-gb/eur/conducting-necessary-condition-analysis-for-business-and-management-students/book262898
Dul, J., van der Laan, E., & Kuik, R. (2020). A statistical significance test for Necessary Condition Analysis. Organizational Research Methods, 23(2), 385-395. doi:10.1177/1094428118795272

See Also

nca_analysis, nca_output

Examples

```
# A more detailed guide can be found here : https://repub.eur.nl/pub/78323/
# or https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2624981

# Load data from a CSV file with header and row names:
data <- read.csv('mydata.csv', row.names=1)
# Or load and rename the example dataset
data(nca.example)
data <- nca.example

# Run NCA with the dataset and name the analysis 'model'.
# Specify the conditions and outcomes by column index or name
# More than 1 condition can be specified with a vector
model <- nca_analysis(data, c(1, 2), 3)

# A quick summary of the analysis can be displayed by 'model'
model

# A full summary of the analysis is shown by nca_output (see documentation for more options)
nca_output(model)

# The results of the analysis is a list of 6 items :
# - plots (1 for each condition)
# - summaries (1 for each condition)
# - bottleneck tables (1 for each ceiling technique)
# - peers (1 dataframe for each condition)
# - tests (1 list for each condition)
# - test.time (total time to run all tests)
names(model)

# The first item contains the graphical outputs for each condition
# This is not really useful to humans
```

```
model$plots[[1]]

# The second item contains a list with the summaries for the conditions
# Extract individual elements with nca_extract
model$summaries[[1]]

# The third item contains a list with the bottleneck tables, one for each ceiling technique
model$bottlenecks$cr_fdh

# The fourth item shows the peers, for each condition
model$peers$Individualism

# For the fifth and sixth item, the test.rep needs to be larger than 0
# for performing the statistical test
# Optionally the p_confidence (default 0.95) and the p_threshold (default 0) can be set
model <- nca_analysis(data, c(1, 2), 3, test.rep=100)

# The fifth item shows the tests for each condition
# This is not really useful to humans
model$tests$Individualism

# The last item shows the total time needed to perform the analysis.
# For large values of test.rep the test may take long.
model$test.time
```

`ceilings` *a set of all available ceiling techniques*

Description

Ceilings to use for the `nca` or `nca_analysis` methods
> nca(data, c(1, 2), 3, ceilings=c('ols', 'ce_fdh', 'cr_fdh'))

Note that the ols regression line is not a ceiling but is included as a reference.

Ceiling Technique	Name
cols	Corrected Ordinary Least Squares
qr	Quantile Regression
ce_vrs	Ceiling Envelopment with Varying Return to Scale
cr_vrs	Ceiling Regression with Varying Return to Scale
ce_fdh	Ceiling Envelopment with Free Disposal Hull
cr_fdh	Ceiling Regression with Free Disposal Hull
c_lp	Ceiling Linear Programming

Note: The SFA and LH ceiling lines are deprecated (discontinued) from version 3.2.0

line.colors *a set defining the line colors for the plots*

Description

Set before calling nca_output
> line.colors['ce_fdh'] <- 'blue'

Reset one line color by setting it to NULL
> line.colors['ce_fdh'] <- NULL

Reset all line colors by setting line.colors to NULL
> line.colors <- NULL

Format

This is a list with default line colors for each ceiling technique

ols	'green'	**c_lp**	'blue'
cols	'darkgreen'	**qr**	'lightpink'
ce_vrs	'orchid4'	**cr_vrs**	'violet'
ce_fdh	'red'	**cr_fdh**	'orange'

line.types *a set defining the line types for the plots*

Description

Set before calling nca_output
> line.types['ce_fdh'] <- 1

Reset one line type by setting it to NULL
> line.types['ce_fdh'] <- NULL

Reset all line types by setting line.types to NULL
> line.types <- NULL

Format

This is a list with default line types for each ceiling technique

ols	1	**c_lp**	2
cols	3	**qr**	4

ce_vrs	5	**cr_vrs**	1
ce_fdh	6	**cr_fdh**	1

`line.width` *parameter defining the width of the lines in the plots*

Description

This will be used for the **lwd** parameter of the plot, default is 1.5.
Set before calling nca_output
> line.width <- 5

`nca` *Run a basic NCA analyses on a data set*

Description

Run a basic NCA analyses on a data set

Usage

```
nca(data, x, y, ceilings=c('ols', 'ce_fdh', 'cr_fdh'))
```

Arguments

`data`	dataframe with columns of the variables
`x`	collection of the columns with the conditions
`y`	index or name of the column with the outcomes
`ceilings`	vector with the ceiling techniques to include in this analysis

Value

Returns a list with 3 items (see examples for further explanation):

`plots`	A list of plot-data for each x-y combination
`summaries`	A list of dataframes with the summaries for each x-y combination
`bottlenecks`	A list of dataframes with a bottleneck table for each ceiling technique

Examples

```
# Load the data
data(nca.example)
data <- nca.example

# Basic NCA analysis
# Conditions in the first 2 columns, outcome in the third column
# This shows scatter plot(s) with the ceiling lines and the effect size(s) on the console
nca(data, c(1, 2), 3)

# Columns can be selected by name as well
nca(data, c('Individualism', 'Risk taking'), 'Innovation performance')

# Define the ceiling techniques via the ceilings parameter
nca(data, c(1, 2), 3, ceilings=c('ols', 'ce_vrs'))
# These are the available ceiling techniques
print(ceilings)
```

---

nca.example *NCA example data with 2 conditions and 1 output*

Description

This data set has Individualism and Risk taking as conditions, and Innovation performance as the output for 28 countries.

Usage

```
data(nca.example)
```

Format

A matrix containing 28 observations, incl. headers and row names.

---

nca.example2 *NCA example data with 3 conditions and 1 outcome*

Description

This data set has Contractual detail, Goodwill trust, and Competence trust as conditions, and Innovation as outcome for 48 buyer-supplier relationships. See Table 2 in: Van der Valk, W., Sumo, R., Dul, J., & Schroeder, R. G. (2016). When are contracts and trust necessary for innovation in buyer-supplier relationships? A necessary condition analysis. Journal of Purchasing and Supply Management, 22(4), 266-277.

Usage

```
data(nca.example2)
```

Format

A matrix containing 48 observations, incl. headers.

`nca_analysis` *Run NCA analyses on a data set*

Description

Run multiple types of NCA analyses on a dataset

Usage

```
nca_analysis(data, x, y, ceilings=c('ols', 'ce_fdh', 'cr_fdh'), custom=NULL,
             corner=NULL, flip.x=FALSE, flip.y=FALSE, scope=NULL,
             bottleneck.x='percentage.range', bottleneck.y='percentage.range',
             steps=10, step.size=NULL, cutoff=0, qr.tau=0.95,
             effect_aggregation = 1, test.rep=0,
             test.p_confidence=0.95, test.p_threshold=0.05)
```

Arguments

`data`	dataframe with columns of the variables
`x`	index or name (or a vector of those) with condition(s) x
`y`	index or name of the column with the outcome y
`ceilings`	vector with the ceiling techniques to include in this analysis
`custom`	custom ceiling line defined by a intercept-slope pair per condition. If only 1 pair given it will be used for all conditions.
`corner`	either an integer or a vector of integers, indicating the corner to analyze, see Details
`flip.x`	reverse the direction of the conditions Use either a boolean for all conditions, or a vector with the same length as x
`flip.y`	reverse the direction of the outcomes, boolean
`scope`	a theoretical scope in list format : (x.low, x.high, y.low, y.high), see Details
`bottleneck.x`	options for displaying the conditions in the bottleneck table 'percentage.range' to display the percentage of range between min(x) and max(x) 'percentage.max' to display the percentage of max(x) 'actual' to display the actual values 'percentile' to display the percentiles Using percentage.max with negative values might yield counterintuitive results.

`bottleneck.y` options for displaying the outcomes in the bottleneck table.
See bottleneck.x

`steps` this argument accepts 2 types :
- an integer (number of steps in the bottleneck table)
- a list of values (used as Y values in the bottleneck table).
Only useful if step.size is not defined.

`step.size` define the step size in the bottleneck table.
The user will be warned if the stepsize does not fit the Y range.
Defaults to null for using steps.

`cutoff` display calculated x,y values that are lower/higher than lowest/highest observed x,y values in the bottleneck table as:
0 : NN (not necessary) and NA (not available)
1 : NN (not necessary) and highest observed values
2 : calculated values

`qr.tau` define the qr tau (between 0 and 1) for the quantile regression ceiling technique, default 0.95

`effect_aggregation`
define the corners to aggregate into the effect size. 1 is upper-left and is always selected, 2 is upper-right, 3 is lower-left and 4 is lower-right

`test.rep` number of resamples in the statistical approximate permutation test. For test.rep = 0 no statistical test is performed

`test.p_confidence`
confidence level of the estimated p-value.
Is used to calculate the p-accuracy for a given number of resamples (test.rep), default 0.95.

`test.p_threshold`
define the threshold significance level in the returned plot of the statistical test, default 0.05

Details

Corners

Corner 1 is the upper-left corner and corner 2 is the upper-right corner. These two corners are used for an analysis of the necessity of the presence/high level if x (corner = 1) or the absence/low level if x (corner = 2) for the presence/high level of y, respectively.
Corner 3 is the lower-left corner and corner 4 is the lower-right corner. These two corners are used for an analysis of the necessity of the presence/high level of x (corner = 3) or the absence/low level if x (corner = 4) for the absence/low level of y, respectively.
By default the upper left corner is analysed for all conditions and corner is not defined. If corner is defined, flip.x and flip.y are ignored.

Scope

By default, the theoretical scope is not defined and the empirical scope is used based on the minimum and maximum observed values of x and y.

Value

Returns a list of 6 items (see examples for further explanation):

plots	A list of plot-data for each x-y combination
summaries	A list of dataframes with the summaries for each x-y combination
bottlenecks	A list of dataframes with a bottleneck table for each ceiling technique
peers	A list of ceilings, with a list of peers for each condition. Peers are corner points of the CE-FDH ceiling line (e.g., the northwest-corners points for corner = 1)
tests	The results of the test for each condition (not human friendly, use nca_output)
test.time	The total time needed to run the tests for all conditions

Examples

```
# Load the data
data(nca.example)
data <- nca.example

# Basic NCA analysis, with conditions in the first 2 columns
# and the outcome in the third column
model <- nca_analysis(data, c(1, 2), 3)

# Use nca_output to show the summaries (see nca_output documentation for more options)
nca_output(model)

# Columns can be selected by name as well
model <- nca_analysis(data, c('Individualism', 'Risk taking'), 'Innovation performance')

# Define the ceiling techniques via the ceilings parameter, see 'ceilings' for all types
model <- nca_analysis(data, c(1, 2), 3, ceilings=c('ce_fdh', 'ce_vrs'))

# These are the available ceiling techniques
print(ceilings)

# Define a custom ceiling line by providing the custom parameter, pairs of intercept-slope
model <- nca_analysis(data, c(1, 2), 3, custom=c(0, 1, 0, 2))

# If a single pair is provided, it will be used for all conditions
model <- nca_analysis(data, c(1, 2), 3, custom=c(0, 1))

# By default, the upper-left corner is analysed. With the corner argument for each
# condition a different corner can be selected. Select corner 1 or 2
# for an analysis of necessary conditions for the presence/high level of the
# outcome, and corner 3 or 4 for an analysis of necessary conditions for
# the absence/low level of the outcome. It is not possible to combine
# corner 1 or 2 with corner 3 or 4 in the same analysis as different outcomes are analysed.
# This analyses the upper right corner for the first condition
# and the upper left corner for the second condition:
model <- nca_analysis(data, c(1, 2), 3, corner=c(2, 1))

# Alternatively, for using the upper right corner(s), 'flip' the x variables
```

```
model <- nca_analysis(data, c(1, 2), 3, flip.x=TRUE)

# It is also possible to flip a single x variable
model <- nca_analysis(data, c(1, 2), 3, flip.x=c(TRUE, FALSE))

# Flip the y variable if the lower corners need analysing
model <- nca_analysis(data, c(1, 2), 3, flip.x=c(TRUE, FALSE), flip.y=TRUE)

# Use a theoretical scope instead of the (calculated) empirical scope
model <- nca_analysis(data, c(1, 2), 3, scope=c(0, 120, 0, 240))

# Display the peers for a ceiling and a condition
print(model$peers$ce_fdh$Individualism)

# By default, the bottleneck tables use percentages of the range for the x and y values.
# Using the percentage of the max value is also possible
model <- nca_analysis(data, c(1, 2), 3, bottleneck.y='percentage.max')

# Use the actual values, in this case the x-value
model <- nca_analysis(data, c(1, 2), 3, bottleneck.x='actual')

# Use percentile, in this case for the y-values
model <- nca_analysis(data, c(1, 2), 3, bottleneck.y='percentile')

# Any combination is possible
model <- nca_analysis(data, c(1, 2), 3, bottleneck.x='actual', bottleneck.y='percentile')

# The number of steps is adjustible via the steps parameter
model <- nca_analysis(data, c(1, 2), 3, steps=20)

# The steps parameter also accepts a list of values
# These are interpreted as actual or percentage / percentile depending on bottleneck.y
model <- nca_analysis(data, c(1, 2), 3, steps=seq(50, 120, 10))

# Or via the step.size parameter, this ignores the steps parameter
model <- nca_analysis(data, c(1, 2), 3, step.size=5)

# If the ceiling line crosses the X = Xmax line at a point C below Y = Ymax,
# for Y < Yc < Ymax, the corresponding X in the bottleneck table is displayed as 'NA'
# It is also possible to display them as Xmax
model <- nca_analysis(data, c(1, 2), 3, cutoff=1)

# or as the calculated value on the ceiling line
model <- nca_analysis(data, c(1, 2), 3, cutoff=2)


# To run tests, the test.rep needs to be larger than 0
# Optionally the p_confidence (default 0.95) and the p_threshold (default 0) can be set
model <- nca_analysis(data, c(1), 3, test.rep=1000,
                      test.p_confidence=0.9, test.p_threshold=0.05)
# The output of the tests can be shown via nca_output with test=TRUE
nca_output(model, test=TRUE)
```

```
# If reproducible outputs are needed, set the seed to a fixed number
set.seed(42)
model <- nca_analysis(data, c(1), 3, test.rep=1000)
```

`nca_difference` *Function for permutation tests*

Description

Function for permutation tests

Usage

```
nca_difference(data1, data2 = NULL, x, y,
ceilings = c("ce_fdh", "cr_fdh"), common_scope, test.rep = 1000,
test.type = c("contrast", "independent", "paired"))
```

Arguments

`data1`	Dataframe 1.
`data2`	Dataframe 2, defaults to NULL.
`x`	Index or name (or a vector of those) with condition(s) x
`y`	Index or name of the column with the outcome y
`ceilings`	Vector with the ceiling techniques to test
`common_scope`	The common scope for both datasets : (x.low, x.high, y.low, y.high), see nca_analysis
`test.rep`	Number of permutations to use, defaults to 1000.
`test.type`	Name of the test type: "contrast", "independent" or "paired".

Examples

```
ceilings <- "cr_fdh"
y <- "Y"
test.rep <- 1000

# Contrast test: effect size difference between two conditions
test.type <- "contrast"
data1 <- nca_random(100, c(0.2, 0.4), c(1,1))
x <- c("X1", "X2")

# For the contrast test, the dataset should be normalized. Use c(0, 1, 0, 1) as common scope
data1 <- nca_util_normalize(data1)
common_scope <- c(0, 1, 0, 1)

difference <- nca_difference(data1 = data1, x = x, y = y,
                             ceilings = ceilings, common_scope = common_scope,
                             test.rep = test.rep, test.type = test.type)
print(difference)
```

```
# Independent test: effect size difference between two datasets
test.type <- "independent"
data1 <- nca_random(100, 0.2, 1)
data2 <- nca_random(100, 0.4, 1)
x <- "X"

# For the independent test, use the (empirical) pooled scope as commn scope
# Or use a theoretical scope, larger then the emperical pooled scope
common_scope <- c(min(data1[, x], data2[, x]), max(data1[, x], data2[, x]),
                  min(data1[, y], data2[, y]), max(data1[, y], data2[, y]))

difference <- nca_difference(data1 = data1, data2 = data2, x = x, y = y,
                             ceilings = ceilings, common_scope = common_scope,
                             test.rep = test.rep, test.type = test.type)
print(difference)

# paired test: effect size differences between 2 measurements
test.type <- "paired"
data1 <- nca_random(100, 0.2, 1)
data2 <- nca_random(100, 0.4, 1)
x <- "X"

# For the paired test, use the (empirical) pooled scope as commn scope
# Or use a theoretical scope, larger then the emperical pooled scope
common_scope <- c(min(data1[, x], data2[, x]), max(data1[, x], data2[, x]),
                  min(data1[, y], data2[, y]), max(data1[, y], data2[, y]))

difference <- nca_difference(data1 = data1, data2 = data2, x = x, y = y,
                             ceilings = ceilings, common_scope = common_scope,
                             test.rep = test.rep, test.type = test.type)
print(difference)
```

`nca_extract` *get model parameters*

Description

Get the model parameters.

Usage

```
nca_extract(model, x=NULL, ceiling=NULL, param='Effect size')
```

Arguments

`model`	The model produced by nca_analysis
`x`	Name of the condition, defaults to the first
`ceiling`	Name of the ceiling, defaults to the first
`param`	Name of the parameter, default 'Effect size'

Examples

```
data(nca.example)
model <- nca_analysis(nca.example, c('Individualism', 'Risk taking'), 3)

# The names of the parameters can be found in the summaries
nca_output(model, plots=FALSE, summaries=TRUE)

# Get the Ceiling zone
extract <- nca_extract(model, 'Individualism', 'ce_fdh', 'Ceiling zone')
print(extract)

# Get the Scope
extract <- nca_extract(model, 'Individualism', 'ce_fdh', 'Scope')
print(extract)

# For multiple values: vectorize one argument and pin the others
names <- c('Individualism', 'Risk taking')
extracts <- sapply(names, nca_extract,
                   model=model, ceiling='ce_fdh', param='Ceiling zone')
print(extracts)

params <- c('Effect size', 'Ceiling zone')
extracts <- sapply(params, nca_extract,
                   model=model, x='Individualism', ceiling='ce_fdh')
print(extracts)

# It is also possible to get the extract directly from the model
# Get the Ceiling zone
print(model$`Ceiling zone`$Individualism$ce_fdh)

# Get the Scope, global parameters are independent of the ceiling
print(model$Scope$Individualism)
```

`nca_outliers` *Outlier detection*

Description

Detect outliers on the dataset.

Usage

```
nca_outliers(data, x, y, ceiling = NULL,
corner = NULL, flip.x = FALSE, flip.y = FALSE, scope=NULL,
k = 1, min.dif = 1e-2, max.results = 25, plotly=FALSE, condensed = FALSE)
```

Arguments

`data`	Dataframe with columns of the variables

x	Index or name of the column with the condition
y	Index or name of the column with the outcome
ceiling	Name of the ceiling technique to be used. If not provided, the default ceilings (CE_FDH) will be used
corner	either an integer or a vector of integers, indicating the corner to analyze, see Details
flip.x	reverse the direction of the conditions Use either a boolean for all conditions, or a vector with the same length as x
flip.y	reverse the direction of the outcomes, boolean
scope	a theoretical scope in list format : (x.low, x.high, y.low, y.high), see nca_analysis
k	use combinations of observations, default is 1 (single observation)
min.dif	set the threshold for the minimum dif.rel to be considered as outlier, default is 1e-2
max.results	only show the first 'max.results' outliers, default is 25
plotly	If true shows the interactive scatter plot(s), one for each condition. In RStudio the plots are shown in the Viewer window. See Details.
condensed	If true and k > 1, hide outlier combinations for which the effect size is not increased compared to the effect size of single elements of that combination. (default FALSE)

Details

Outliers

The potential outliers are displayed with the original effects size and the effect size if this outlier is removed. The absolute and relative differences between both effect sizes is also shown. The table also displays if a point is a ceiling zone outlier or a scope outlier.

Plotly

The plot highlights the potential outliers. The names, relative difference and XY coordinates of all points pop up when moving the pointer over the plot. The toolbar allows several actions such as zoom and selection of parts of the plot.

Examples

```
# A basic example of the nca_outliers command:
data(nca.example)
outliers <- nca_outliers(nca.example, 1, 3)

# This prints the outlier table
print(outliers)

# Plotly displays a scatterplot with the outliers
nca_outliers(nca.example, 1, 3, plotly = TRUE)

# Test for combinations of observations
```

```
# Useful to detect clusters of observations as possible outliers
nca_outliers(nca.example, 1, 3, k = 2)

# Just like the nca_analysis command, nca_outliers accept both flip and corner arguments
nca_outliers(nca.example, 1, 3, corner=3)

# It is possible to define the maximum number of results (default is 25)
nca_outliers(nca.example, 1, 3, max.results=5)

# Do no show possible outliers where the abs(dif.rel) is smaller than min.dif
nca_outliers(nca.example, 1, 3, min.dif=10)

# If k > 1, the effect size of a single observation might not change
# when paired with another observation, e.g. dif.rel of Obs1 == dif.rel of Obs1+Obs2.
# The example below hides combinations of Japan with Portugal, Greece, etc.
nca_outliers(nca.example, 1, 3, k = 2, condensed = TRUE)
```

`nca_output` *display the result of the NCA analysis*

Description

Show the plots, NCA summaries and bottleneck tables of a NCA analysis.

Usage

```
nca_output(model, plots=TRUE, plotly=FALSE, bottlenecks=FALSE,
           summaries=TRUE, test=FALSE, pdf=FALSE, path=NULL, selection = NULL,
           plot_bottlenecks=0)
```

Arguments

`model`	Displays output of the nca or nca_analysis command
`plots`	If true (default) show the scatter plot(s), one for each condition. In Rstudio the plots are shown the Plots window.
`plotly`	If true shows the interactive scatter plot(s), one for each condition. In RStudio the plots are shown in the Viewer window. See Details.
`bottlenecks`	If true displays the bottleneck table(s) in the Console window, one table for each ceiling line
`summaries`	If true shows the summaries for each condition in the Console window, see Details
`test`	If true shows the result of the statistical sigificance test (if present), see Details
`pdf`	If true exports the output to a pdf file, except for the plotly plot
`path`	Optional path for the output file(s)

`selection` Optionally selects the conditions for inclusion in the output.
For example, only the plots for the selected condition are shown.
In the bottleneck table the required levels of only the selected conditions for given levels of Y are shown.

`plot_bottlenecks`
Optionally show the bottleneck lines

Details

Plotly

The plot highlights the points that construct the ceiling line ('peers'). The names and XY coordinates of all points pop up when moving the pointer over the plot. The toolbar allows several actions such as zoom and selection of parts of the plot. Optionally subgroups of points can be labeled.

Summaries

The output starts with 6 lines of basic information ("global") about the dataset ("Number of observations", "Scope", "Xmin", "Xmax", "Ymin", and "Ymax"). "Scope" refers to the empirical area of possible x-y combinations, given the minimum and maximum observed x and y values.
The next 11 lines present the NCA parameters ("param", see below) for each of the selected ceiling techniques (the defaults techniques are ce_fdh and cr_fdh).

The 11 printed NCA parameters are:
- *Ceiling zone*, which is the size of the "empty" area in the upper-left corner
- *Effect size*, which is the ceiling zone divided by the scope
- *# above*, which is the number of observations that are above the ceiling line and hence in the "empty" ceiling zone
- *Ceiling accuracy*, which is the number of observations on or below the ceiling line divided by the total number of observations and multiplied by 100 percent
- *Fit*, which relates to the "closeness" of the selected ceiling line to the ce_fdh ceiling line
- *Slope* and *Intercept*, which are the slope and the intercept of the straight ceiling line (no values are printed if the ceiling line is not a straight line, but a step function)
- *Abs. ineff.*, which is the total xy-space where x does not constrain y, and y is not constrained by x
- *Rel. ineff.*, which is the total xy-space where x does not constrain y, and y is not constrained by x as percentage of the scope
- *Condition ineff.*, which is the condition inefficiency that indicates for which range of x (as a percentage of the total range) x does not constrain y (i.e., there is no ceiling line in that x-range)
- *Outcome ineff.*, which is the outcome efficiency that indicates for which range of y (as a percentage of the total range of y) y is not constrained by x (i.e., there is no ceiling line in that y-range)

Test

NCA's statistical test is a randomness test to evaluate if the observed effect size may be a random result of unrelated X and Y variables

Examples

```
# Use the result of the nca command:
data(nca.example)
data <- nca.example
model <- nca_analysis(data, c(1, 2), 3)
```

```
# Show the full summaries in the Console window
nca_output(model)

# Suppress the summaries and display the plots
nca_output(model, plots=TRUE, summaries=FALSE)

# Display the plots via Plotly
nca_output(model, plotly=TRUE, summaries=FALSE)

# Label the observation of the Plotly plot by using a vector of names (no more than 5).
# For example label the observations in nca.example
labels <- c('Australia', 'Europe', 'Europe', 'North America', 'Europe', 'Europe',
'Europe', 'Europe', 'Europe', 'Europe', 'Europe', 'Europe', 'Asia',
'North America', 'Europe', 'Australia', 'Europe', 'Europe', 'Europe', 'Europe',
'Asia', 'Europe', 'Europe', 'Europe', 'Europe', 'Europe', 'North America')
nca_output(model, plotly=labels, summaries=FALSE)

# Suppress the summaries and display the bottlenecks
nca_output(model, bottlenecks=TRUE, summaries=FALSE)

# Show the results of the statistical significance test (p-value)
# Make sure to set test.rep in nca_analysis
nca_output(model, test=TRUE)

# Show all five
nca_output(model, plots=TRUE, plotly=TRUE, bottlenecks=TRUE, test=TRUE)

# Per condition, export plots and summaries to PDF files,
# and export all the bottleneck tables to a single PDF file
nca_output(model, plots=TRUE, bottlenecks=TRUE, pdf=TRUE)

# Use the path option to export to an existing directory
outdir <- '/tmp'
nca_output(model, plots=TRUE, pdf=TRUE, path=outdir)

# Limit the output to a selection of conditions by name
nca_output(model, plots=TRUE, selection=c("Individualism"))

# Or by column index, in both cases the order matters
nca_output(model, plots=TRUE, selection=c(2, 1))

# Show the bottleneck lines in the plots or plotly
# 1 shows the outcome lines, 2 also shows the lines for the condition
nca_output(model, plots = TRUE, bottlenecks = TRUE, summaries = FALSE, plot_bottlenecks = 1)
```

`nca_power` *Function to evaluate power*

Description

Function to evaluate power, test if a sample size is large enough to detect necessity.

Usage

```
nca_power(n = c(20, 50, 100), effect = 0.10, slope = 1, ceiling = "ce_fdh",
corner = 1, p = 0.05, distribution.x = "uniform", distribution.y = "uniform",
rep = 100, test.rep = 200)
```

Arguments

n	Number of datapoints to generate, either an integer or a vector of integers.
effect	Effect size of the generated datasets (single value or vector of values).
slope	Slope of the line (single value or vector of values).
ceiling	Ceiling technique to use for this analysis (single value or vector of values).
corner	Integer, indicating the corner to analyze, see nca_analysis.
p	Significance level.
distribution.x	Distribution type(s) for X, "uniform" (default) or "normal" (single value or vector of values).
distribution.y	Distribution type(s) for Y, "uniform" (default) or "normal" (single value or vector of values).
rep	Number of analyses done per iteration.
test.rep	Number of resamples in the statistical approximate permutation test. For test.rep = 0 no statistical test is performed. See nca_analysis for reproducible outputs.

Examples

```
# Simple example
## Not run: results <- nca_power()

print(results)
```

---

nca_powerplot *Function to show the powerplot*

Description

Function to show the powerplot, optional over various variables.

Usage

```
nca_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	Dataframe as returned by the nca_power function.
x.variable	The variable to use for the X scale, default is sample size.
x.name	The name for the X scale.
x.min	The minimum value for the X scale, defaults to 0.
x.max	The maximum value for the X scale, defaults to the maximum X.
line.variable	Variable that represented the line.
line.name	The name of the variable that represented the line.

Examples

```
# Simple example, showing a normal plot with the power as a function of the sample size
# A line for each effect size will be shown
## Not run: results <- nca_power(effect = c(.1, .2, .3))

nca_powerplot(results)

# This example shows the power for different ceiling lines
## Not run: results <- nca_power(ceiling = c("ce_fdh", "cr_fdh"))

powerplot <- nca_powerplot(results, line.variable='ceiling', line.name='Ceiling')
```

---

nca_random *generating random data that meets necessity*

Description

Generate N datapoints, with 'normal' or 'uniform' distributions for X and Y

Usage

```
nca_random(n, intercepts, slopes, corner=1,
                  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 to generate, should be an integer > 1.
intercepts	The intercept or a vector of intercepts of the line.
slopes	The slope or a vector if slopes of the line.
corner	Define which corner should be empty, default is 1 (upper left).
distribution.x	Type of the distribution for X, "uniform" (default) or "normal". The latter is a truncated normal distribution.
distribution.y	Type of the distribution for Y, "uniform" (default) or "normal". The latter is a truncated normal distribution.

mean.x	Distribution Mean of X (default 0.5), ignored distribution.x == "uniform".
mean.y	Distribution Mean of Y (default 0.5), ignored distribution.y == "uniform".
sd.x	Distribution SD of X (default 0.2), ignored distribution.x == "uniform".
sd.y	Distribution SD of Y (default 0.2), ignored distribution.y == "uniform".

Examples

```
# Generate a uniform dataset, default for X and Y
data <- nca_random(100, 0, 1)

# It is also possible to generate a dataset with multiple conditions,
# by supplying vectors for the intercepts and slopes
data <- nca_random(100, c(0, 0.25), c(1, 0.75))
# Single values will be repeated to complement a vector
data <- nca_random(100, c(0, 0.25), 1)

# The default is an empty space in the upper left corner.
# A different corner can be selected with the corner argument
data <- nca_random(100, 0, 1, corner=4)

# Generate a dataset with a normal distribution for X and a uniform distribution for Y
data <- nca_random(100, 0, 1, distribution.x = "normal", distribution.y = "uniform")

# Generate a dataset with a normal distribution for X and Y, with adjusted MEAN
data <- nca_random(100, 0, 1, distribution.x = "normal", distribution.y = "normal",
                   mean.x = 0.75, mean.y = 0.75)

# Generate a dataset with a normal distribution for X and Y, with adjusted SD
data <- nca_random(100, 0, 1, distribution.x = "normal",
                   distribution.y = "normal", sd.x = 0.1, sd.y = 0.1)
```

---

nca_util_normalize *Normalizes numeric data*

Description

(Min-max) Normalize the columns of the data

Usage

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

Arguments

data	Input dataframe or matrix.
scale	Set(s) of output Min-Max values.
min_max	Set(s) of (theoretical) Min-Max values.

Details

scale
If scale is NULL (default), all columns will be normalized between 0 and 1.
Use c(0, 100) to normalize between 0 and 100.
Replicate for each column for different scale, e.g. c(0, 1, 0, 1, 0, 100).
Use NA in the min-max pair if a column does not need to be normalized.

Examples

```
# Normalize the data between 0 and 1
data(nca.example)
data.normalized <- nca_util_normalize(nca.example)
# Equivalent to this
data.normalized <- nca_util_normalize(nca.example, c(0, 1))

# Normalize the data between 0 and 100
data.normalized <- nca_util_normalize(nca.example, c(0, 100))

# Normalize each column with different values
data.normalized <- nca_util_normalize(nca.example, c(0, 1, 0, 10, 0, 100))

# Use NA if a column doesn't need to be normalized
data.normalized <- nca_util_normalize(nca.example, c(0, 1, NA, NA, 0, 100))

# It is also possible to use a theoretic range for the normalization
data.normalized <- nca_util_normalize(nca.example, min_max = c(0, 100, 0, 150, 0, 300))
```

---

point.color *parameter defining the point color in the plots*

Description

This will be used for the **col** parameters of the plots, default is blue.
Set before calling nca_output
> point.color <- red

---

point.type *parameter defining the plotting symbol in the plots*

Description

This will be used for the **pch** parameter of the plots, default is 21.
Set before calling nca_output
> point.type <- 22

See http://www.sthda.com/english/wiki/r-plot-pch-symbols-the-different-point-shapes-available-in-r for more symbols

Index