Package ‘SetMethods’

July 21, 2025

Type Package

Title Functions for Set-Theoretic Multi-Method Research and Advanced QCA

Version 4.1

Date 2025-03-21

Maintainer Ioana-Elena Oana <`ioana.oana@eui.eu`>

Description
Functions for performing set-theoretic multi-method research, QCA for clustered data, theory evaluation, Enhanced Standard Analysis, indirect calibration, radar visualisations. Additionally it includes data to replicate the examples in the books by Oana, I.E, C. Q. Schneider, and E. Thomann. Qualitative Comparative Analysis (QCA) using R: A Beginner's Guide. Cambridge University Press and C. Q. Schneider and C. Wagemann ``Set Theoretic Methods for the Social Sciences'', Cambridge University Press.

Depends QCA, admisc, methods, ggplot2, ggrepel, stargazer, R (>= 3.5.0)

Imports scatterplot3d, fmsb, betareg

License GPL-2

NeedsCompilation no

Date/Publication 2025-03-21 14:50:02 UTC

Author Ioana-Elena Oana [aut, cre],
Juraj Medzihorsky [aut],
Mario Quaranta [aut],
Carsten Q. Schneider [aut]

Repository CRAN

Contents

`SetMethods-package` *Functions for Set-Theoretic Multi-Method Research and Advanced QCA*

Description

This initiated as a package companion to the book by C. Q. Schneider and C. Wagemann "Set-Theoretic Methods for the Social Sciences", Cambridge University Press. It has now grown to include functions for performing set-theoretic multi-method research (Schneider 2023, Schneider and Rohlfing 2013), QCA for clustered data (Garcia-Castro and Arino 2013), theory evaluation, Enhanced Standard Analysis, QCA Radar Charts, indirect calibration, robustness tests (Oana and Schneider - 2021) etc.. Additionally it includes data to replicate the examples in the book by Oana, I.E, C. Q. Schneider, and E. Thomann. Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge University Press and C. Q. Schneider and C. Wagemann "Set Theoretic Methods for the Social Sciences", 2021, Cambridge University Press.

Details

Package:	SetMethods
Type:	Package
Version:	4.1
Date:	2025-03-21
License:	GPL-2

The package contains functions to perform set-theoretic multi-method research, theory evaluation, QCA for clustered data, Enhanced Standard Analyses, indirect calibration, calculate parameters of fit and produce XY plots and QCA Radar Charts, perform robustness tests, etc.. Furthermore, it contains all the data used in the Schneider and Wagemann (2012) and Oana, Schneider, and Thomann (2021) books.

Author(s)

Ioana-Elena Oana [aut, cre], Juraj Medzihorsky [aut], Carsten Q. Schneider [aut], Mario Quaranta [aut]

Maintainer: Ioana-Elena Oana <ioana.oana@eui.eu>

References

Schneider, C. Q. (2023). Set-Theoretic Multi-Method Research: A Guide to Combining QCA and Case Studies. Cambridge: Cambridge University Press.

Oana, I.E., Schneider, C. Q., Thomann, E. (2021). Qualitative Comparative Analysis (QCA) using R: A Beginner's Guide. Cambridge: Cambridge University Press.

Oana, I.E., Schneider, C. Q. (2021). A Robustness Test Protocol for Applied QCA: Theory and R Software Application. Sociological Methods and Research. https://doi.org/10.1177/00491241211036158.

Oana, I.E., Schndeider, C.Q. (2018) SetMethods: An Add-on R Package for Advanced QCA.The R Journal 10(1): 507-33.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge

Schneider, C. Q., Rohlfing, I. (2013) Combining QCA and Process Tracing in Set-Theoretic Multi-Method Research. Sociological Methods Research 42(4): 559-597

Haesebrouck, T. (2015) Pitfalls in QCA's consistency measure. Journal of Comparative Politics 2:65-80.

Garcia-Castro, A., Arino, M. A.. (2013) A General Approach to Panel Data Set-Theoretic Research. COMPASSS Working Paper 2013-76

`ambig.cases` *Function for identifying cases with 0.5 fuzzy-set values.*

Description

A function that identifies cases with 0.5 fuzzy-set values.

Usage

```
ambig.cases(data)
```

Arguments

`data`	A datafarme, a subset of a dataframe, or a vector (i.e. single column in a dataframe). If the function is provided with the dataframe it will return the name of the cases with 0.5 values together with their location in the dataframe. If the function is provided with a vector (i.e. a single column), it will return the position of the case having a 0.5 in that vector. The function should be used for calibrated data and will give an error if the data contains uncalibrated scores. However, if you have both calibrated and uncalibrated data in the same dataframe, it is possible to use the function only for the calibrated subset of that data.

Author(s)

Ioana-Elena Oana

Examples

```
# Import your data. For example:

data(SCHF)

# Get cases with 0.5 in the entire dataframe:

ambig.cases(SCHF)
```

```
# Get cases with 0.5 in the column "EMP" in the dataframe:

ambig.cases(SCHF$EMP)

# Get cases with 0.5 in the 7th column of the dataframe:

ambig.cases(SCHF[,7])
```

BCMV *Berg-Schlosser and Cronqvist (2005)*

Description

The `BCMV` data frame has 18 rows and 5 variables

Usage

```
data(BCMV)
```

Format

A data frame with 18 observations on the following 5 variables.

`GNP` a numeric vector. Condition, Gross National Product/Capita (ca. 1930). 0 if below 500$, 1 if between 550 and 850$, 2 above 850$.

`URB` a numeric vector. Condition, urbanization (population in towns with 20000 and more inhabitants); 0 if below 50 per cent; 1 if above.

`LIT` a numeric vector. Condition, literacy: 0 if below 75 per cent; 1 if above.

`INDUS` a numeric vector. Condition, Industrial Labour Force (incl. mining); 0 if below 30 per cent of active population; 1 if above.

`DEMOC` a numeric vector. Condition, stability of a democracy: 0 if not stable; 1 if stable.

Details

The data are used by Berg-Schlosser and Cronqvist (2005) to demostrate mvQCA. The original data are from Lipset (1963). Data are multi-value.

References

Berg-Schlosser, D. and Cronqvist, L. (2005) "Macro-Quantitative vs. Macro-Qualitative Methods in the Social Sciences - An Example from Empirical Democratic Theory", Historical Social Research 30, pp. 154-175.

Lipset, Seymour M. (1963) Political Man. The Social Bases of Politics. Doubleday: New York.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Schneider, C. Q., Wagemann, C., Quaranta, M. (2012) How To... Use Software for Set-Theoretic Analysis. Online Appendix to "Set-Theoretic Methods for the Social Sciences". Available at www.cambridge.org/schneider-wagemann

Examples

```
data(BCMV)
```

`cluster` *Diagnostic tool for clustered data.*

Description

Function returns pooled, within, and between consistencies for the relationship between two sets, for an object of class "qca", and for a Boolean expression.

Usage

```
cluster(data=NULL, results, outcome,
      unit_id, cluster_id, sol = 1,
      necessity = FALSE, wicons = FALSE)
```

Arguments

`data`	A data frame in the long format containing both a colum with the unit names and a column with the cluster names. Column names should be in capital letters.
`results`	An object of class "qca". For performing cluster diagnostics of the sufficient solution for the negated outcome one must only use the `minimize()` result from the sufficiency analysis of the negated outcome. The argument results can also be a vector, a character string, or a boolean expression of the form e.g. "A*~B + ~B*C".
`outcome`	A character string with the name of the outcome in capital letters. When performing cluster diagnostics of the sufficient solution for the negated outcome one must only use the `minimize()` result from the sufficiency analysis of the negated outcome in the argument `results`. When performing cluster diagnostics for boolean expressions or vectors the negated outcome can be used by inserting a tilde in the outcome name in the argument `outcome`. The outcome can also be a vector.
`unit_id`	A character string with the name of the vector containing the units (e.g. countries).
`cluster_id`	A character string with the name of the vector containing the clustering units (e.g. years).
`sol`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.

`necessity`	Logical. Perform the diagnostic for the relationship of necessity?
`wicons`	Logical. Should within consistencies and coverages be printed?

Author(s)

Ioana-Elena Oana

References

Garcia-Castro, Roberto, and Miguel A. Arino. 2016. "A General Approach to Panel Data Set-Theoretic Research."" Journal of Advances in Management Sciences & Information Systems 2: 6376.

See Also

`minimize`

Examples

```
# Import your clustered data in the long format.
# For example:

data(SCHLF)

# Get the intermediate solution:

sol_yi <- minimize(SCHLF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE, dir.exp = c(0,0,0,0,0,0))

# Get pooled, within, and between consistencies for the intermediate solution:

cluster(SCHLF, sol_yi, "EXPORT", unit_id = "COUNTRY",
              cluster_id = "YEAR", sol = 1)

# or:

cluster(SCHLF, sol_yi, "EXPORT", unit_id = "COUNTRY",
              cluster_id = "YEAR", sol = "c1p1i1")

# Get pooled, within, and between consistencies for EMP as necessary for EXPORT:

cluster(SCHLF, results="EMP", outcome="EXPORT", unit_id = "COUNTRY",
              cluster_id = "YEAR", necessity=TRUE)
# or:

cluster(results=SCHLF$EMP, outcome=SCHLF$EXPORT, unit_id = SCHLF$COUNTRY,
              cluster_id = SCHLF$YEAR, necessity=TRUE)

# Get pooled, within, and between consistencies for ~EMP as necessary for EXPORT:
```

```
cluster(SCHLF, results="~EMP", outcome="EXPORT", unit_id = "COUNTRY",
              cluster_id = "YEAR", necessity=TRUE)
# or:

cluster(results=1-SCHLF$EMP, outcome=SCHLF$EXPORT, unit_id = SCHLF$COUNTRY,
              cluster_id = SCHLF$YEAR, necessity=TRUE)

# Get pooled, within, and between consistencies for EMP*~MA*STOCK as sufficient for EXPORT:

cluster(SCHLF, "EMP*~MA*STOCK", "EXPORT", unit_id = "COUNTRY",
              cluster_id = "YEAR")

# Get pooled, within, and between consistencies for EMP*MA + ~STOCK as sufficient for ~EXPORT:

cluster(SCHLF, "EMP*MA + ~STOCK", "~EXPORT", unit_id = "COUNTRY",
              cluster_id = "YEAR")
```

`cluster.plot` *Function for plotting pooled, between, and within consistencies for a cluster diagnostics.*

Description

Function for plotting pooled, between, and within consistencies for a cluster diagnostics. For a sufficient solution, the function returns plots for the entire solution and each sufficient term.

Usage

```
cluster.plot(cluster.res,
            labs = TRUE,
            size = 5,
            angle = 0,
            wicons = FALSE,
            wiconslabs = FALSE,
            wiconssize = 5,
            wiconsangle = 90)
```

Arguments

`cluster.res`	An object of class "cluster.res". This is the result of a cluster diagnostics obtained with the cluster() function.
`labs`	Logical. Should labels for the clusters be printed?
`size`	Label font size for the clusters.
`angle`	Label rotation for the clusters.
`wicons`	Logical. Should within consistency plots be returned?
`wiconslabs`	Logical. Should labels for the units be printed?
`wiconssize`	Label font size for the units.
`wiconsangle`	Label rotation for the units.

Author(s)

Ioana-Elena Oana

References

Garcia-Castro, Roberto, and Miguel A. Arino. 2016. "A General Approach to Panel Data Set-Theoretic Research."" Journal of Advances in Management Sciences & Information Systems 2: 6376.

Examples

```
# Load the data:
data(PAYF)

# Create a sufficient solution using minimize:
PS <- minimize(data = PAYF,
                        outcome  = "HL",
                        conditions = c("HE","GG","AH","HI","HW"),
                        incl.cut = 0.9,
                        n.cut = 2,
                        include = "?",
                        details = TRUE,
                        show.cases = TRUE)
PS

# Perform cluster diagnostics:

CB <- cluster(data = PAYF,
           results = PS,
           outcome = "HL",
           unit_id = "COUNTRY",
           cluster_id = "REGION",
           necessity=FALSE,
           wicons = FALSE)
CB

# Plot pooled, between, and within consistencies:

cluster.plot(cluster.res = CB,
             labs = TRUE,
             size = 8,
             angle = 6,
             wicons = TRUE)
```

`EMMF` *EMMFnegger (2011)*

Description

The `EMMF` data frame has 19 rows and 8 sets

Usage

```
data(EMMF)
```

Format

A data frame with 19 observations on the following 8 sets.

`country` a factor with levels `Australia` `Austria` `Belgium` `Canada` `Denmark` `Finland` `France` `Germany` `Ireland` `Italy` `Netherlands` `NewZealand` `Norway` `Portugal` `Spain` `Sweden` `Switzerland` `UK` `USA`

`s` a numeric vector. Condition, state-society relationships.

`c` a numeric vector. Condition, non-market coordination.

`l` a numeric vector. Condition, strength of the labour movement.

`r` a numeric vector. Condition, religious denomination.

`p` a numeric vector. Condition, strenght of religious parties.

`v` a numeric vector. Condition, institutional veto points.

`jsr` a numeric vector. Outcome, job-sectурity regulations.

Details

Data are used by Emmenegger (2011) to analyze job-security regulations in Western democracies. The data are fuzzy-sets.

References

Emmenegger, P. (2011) "Job-security regulations in Western democracies", European Journal of Political Research 50, pp. 336-364.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(EMMF)
```

esa *Function that performs the Enhanced Standard Analysis.*

Description

Function that performs the Enhanced Standard Analysis.

Usage

```
esa(oldtt, nec_cond, untenable_LR, contrad_rows)
```

Arguments

`oldtt`	A truthTable object.
`nec_cond`	A vector of character strings containing the necessary conditions. Conditions should be capitalized and negated conditions should be inserted with a "~". Unions of conditions are performed with a "+". Using this argument, logical remainder rows that contradict the statement of necessity will not be used in the analysis (i.e. OUT will be set to 0 in the truth table).
`untenable_LR`	A Boolean expression containing the untenable logical remainders. Conditions should be capitalized and negated conditions should be inserted with a "~". Intersections of conditions are performed with a "*". Using this argument, logical remainder rows containing the particular intersection specified will not be used in the analysis (i.e. OUT will be set to 0 in the truth table).
`contrad_rows`	A vector containing the names of the rows that are contradictory. Using this argument, all rows with the names specified (both logical remainders and rows containing empirical information) will not be used in the analysis (i.e. OUT will be set to 0 in the truth table).

Value

It returns a new truth table in which all truth table rows are set to outcome value 0 that would otherwise present untenable assumptions.

Author(s)

Ioana-Elena Oana

References

Schneider, C. Q., Wagemann, C. 2012. Set-Theoretic Methods for the Social Sciences: A Guide to Qualitative Comparative Analysis. Cambridge: Cambridge University Press, chapter 8.

See Also

`minimize`

Examples

```
# Import your data. For example:

data(SCHF)

# Get the truth table for the presence of the outcome:

TT_y <- truthTable(SCHF, outcome = "EXPORT",
                   conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                   incl.cut = .9,
                   complete = TRUE,
                   PRI = TRUE,
                   sort.by = c("out", "incl", "n"))

# Exclude condition STOCK + MA and condition EMP as necessary for EXPORT
# Exclude all remainder rows containing the combination BARGAIN*~OCCUP
# Exclude the rows "19", "14", "46", "51" as contradictory:

newtt <- esa(oldtt = TT_y, nec_cond = c("STOCK+MA", "EMP"),
          untenable_LR = "BARGAIN*~OCCUP", contrad_rows = c("19", "14", "46", "51"))

# The truth table newly created can afterwards be used in further analyses
```

FakeCS *Fake crisp-set data*

Description

The FakeCS data frame has 30 rows and 5 sets

Usage

```
data(FakeCS)
```

Format

A data frame with 30 observations on the following 5 sets.

y a numeric vector. Outcome with 2 categories (crisp-set).

j a numeric vector. Condition with 2 categories (crisp-set).

z a numeric vector. Condition with 2 categories (crisp-set).

w a numeric vector. Condition with 2 categories (crisp-set).

k a numeric vector. Condition with 2 categories (crisp-set).

Details

The data frame has only exercise purpuses to let the user learn how to perform crisp-set QCA in R.

References

Schneider, C. Q., Wagemann, C., Quaranta, M. (2012) How To... Use Software for Set-Theoretic Analysis. Online Appendix to "Set-Theoretic Methods for the Social Sciences". Available at www.cambridge.org/schneider-wagemann.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(FakeCS)
```

`FakeMV` *Fake data for mvQCA*

Description

`mvQCA` data frame has 25 rows and 4 sets.

Usage

```
data(FakeMV)
```

Format

A data frame with 25 observations on the following 4 sets.

`Y` a numeric vector. Outcome with 2 categories (crisp).

`A` a numeric vector. Condition with 2 categories (crisp).

`B` a numeric vector. Condition with 3 categories (multi-value).

`C` a numeric vector. Condition with 3 categories (multi-value).

Details

The data frame has only exercise purpuses to let the user learn how to perform mvQCA in R.

References

Schneider, C. Q., Wagemann, C., Quaranta, M. (2012) How To... Use Software for Set-Theoretic Analysis. Online Appendix to "Set-Theoretic Methods for the Social Sciences". Available at www.cambridge.org/schneider-wagemann.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(FakeMV)
```

FSR *Freitag and Schlicht (2009)*

Description

The FSR data frame has 16 rows and 8 sets

Usage

```
data(FSR)
```

Format

A data frame with 16 observations on the following 8 sets.

`integrated_comp_schools` a numeric vector. Condition, percentage of Pupils Enrolled in Integrated Comprehensive Schools,

`coop_comp_schools` a numeric vector. Condition, percentage of Pupils Enrolled in Cooperative Comprehensive Schools.

`full_day_schools` a numeric vector. Condition, percentage of Pupils Enrolled in All-Day Schools

`child_care` a numeric vector. Condition, ratio of Number of Child Care Facilities to Total Population between 0 and 6 Years (percent).

`pre_schools` a numeric vector. Condition, ratio of pupils Enrolled in Pre-School to Total 6-Year-Old Population (per cent)

`early_tracking` a numeric vector. Condition, onset of Tracking, Legal Regulation.

`outcome` a numeric vector. Outcome, high Degree of Social Inequality Cases in Education.

`indep_hauptschule` a numeric vector. Condition, autonomy of the Hauptschule.

Details

Data are used by Freitag and Schlicht (2009) to analyze social inequality in education. The data are raw scores.

References

Freitag, M, and Schlicht, R. (2009) "Educational Federalism in Germany: Foundations of Social Inequalities in Education", Governance 22(1), pp. 47-72.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(FSR)
```

`indirectCalibration` *Function performing the indirect calibration*

Description

indirectCalibration is a function for the indirect calibration procedure as described by Ragin (2008). It uses a binomial or a beta regression for tranforming raw scores into calibrated scores. In our opinion, using a fractional polynomial may not be appropriate to this case. In fact, we do not deal with proportions. This function requires the package `betareg`.

Usage

```
indirectCalibration(x, x_cal, binom = TRUE)
```

Arguments

`x`	vector of raw scores.
`x_cal`	vector of theoretically calibrated scores.
`binom`	logical. If indirect calibration has to be performed using binomial regression or beta regression. The default is `TRUE`, which means that binomial regression is used.

Value

It returns a vector of indirectly calibrated values.

Author(s)

Mario Quaranta

References

Ragin, C. C. (2008) Redesigning Social Inquiry: Fuzzy Sets and Beyond, The Chicago University Press: Chicago and London.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
# Generate fake data
set.seed(4)
x <- runif(20, 0, 1)

# Find quantiles
quant <- quantile(x, c(.2, .4, .5, .6, .8))

# Theoretical calibration
x_cal <- NA
```

```
x_cal[x <= quant[1]] <- 0
x_cal[x > quant[1] & x <= quant[2]] <- .2
x_cal[x > quant[2] & x <= quant[3]] <- .4
x_cal[x > quant[3] & x <= quant[4]] <- .6
x_cal[x > quant[4] & x <= quant[5]] <- .8
x_cal[x > quant[5]] <- 1
x_cal

# Indirect calibration (binomial)
a <- indirectCalibration(x, x_cal, binom = TRUE)

# Indirect calibration (beta regression)
b <- indirectCalibration(x, x_cal, binom = FALSE)

# Correlation
cor(a, b)

# Plot
plot(x, a); points(x, b, col = "red")
```

intersectExp *Intersects two boolean expressions.*

Description

Function that intersects two boolean expressions.

Usage

```
intersectExp(expression1, expression2)
```

Arguments

expression1	A boolean expression. Conditions should be capitalized and negated conditions should be inserted with a "~". Unions of conditions are performed with a "+", while intersections are performed with a "*".
expression2	A boolean expression. Conditions should be capitalized and negated conditions should be inserted with a "~". Unions of conditions are performed with a "+", while intersections are performed with a "*".

Value

It returns the boolean expression representing the intersection of the two inputed expressions.

Author(s)

Ioana-Elena Oana

Examples

```
intersectExp("~EMP*MA", "MA+~STOCK*OCCUP")
intersectExp("~A*B + C*~D","A*B+~D")
```

`JOBF` *Fake fuzzy-set job motivation data.*

Description

Fake fuzzy-set job motivation data used for Chapter 3 of the Oana et al. (forthcoming) book.

Usage

```
data(JOBF)
```

Format

A data frame with 17 observations on the following 3 variables.

`HS` Condition: High salary

`FS` Condition: Flexible schedule

`LS` Condition: Lenient supervisor

`LJS` Condition: Low job security

`PP` Condition: High promotion potential

`HM` Outcome: High motivation

`FSorLS` Condition:

References

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
data(JOBF)
```

KAF *Koenig-Archibugi (2004)*

Description

The KAF data frame has 13 rows and 5 sets

Usage

```
data(KAF)
```

Format

A data frame with 13 observations on the following 5 sets.

`supranat`	a numeric vector. Government support for supranational CFSP.
`identmass`	a numeric vector. European identity of the general public.
`conform`	a numeric vector. Policy conformity.
`region`	a numeric vector. Regional governance.
`capab`	a numeric vector. Material capabilities.

Details

Data are used by Koenig-Archibugi (2004) to analyze government preferences for istitutional change in EU foreign and security policy. Data are fuzzy-sets.

References

Koenig-Archibugi, M. (2004) "Explaining Government Preferences for Istitutional Change in EU Foreign and Security Policy", International Organization 58, pp.137-174.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(KAF)
```

`LIPC` *Lipset (1959), crisp-set*

Description

The `LIPC` data frame has 18 rows and 6 sets

Usage

```
data(LIPC)
```

Format

A data frame with 18 observations on the following 6 sets.

`DEVELOPED` a numeric vector. Condition, economically developed country.

`URBAN` a numeric vector. Condition, urbanized countries.

`LITERATE` a numeric vector. Condition, countries with high literacy rate.

`INDUSTRIAL` a numeric vector. Condition, Industrialized countries.

`GOVSTAB` a numeric vector. Condition, politically stable countries.

`SURVIVED` a numeric vector. Outcome, survival of democracy during the inter-war period.

Details

Data used by Ragin (2009) to illustrates the variants of QCA. Originally by Lipset (1959). Data are crisp-sets.

References

Lipset, S. M. (1959) "Some Social Requisites of Democracy: Economic Development and Political Legitimacy", American Political Science Review 53, pp. 69-105.

Ragin, C. C. (2009) "Qualitative Comparative Analysis. Using Fuzzy Sets (fsQCA)." In Rihoux, B., and Ragin, C. C. (eds.) Configurational Comparative Methods. Qualitative Comparative Analysis (QCA) and Related Techniques. Thousand Oaks, CA and London: Sage, pp. 87-121.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(LIPC)
```

LIPF *Lipset (1959), fuzzy-set*

Description

The LIPF data frame has 18 rows and 6 sets

Usage

```
data(LIPF)
```

Format

A data frame with 18 observations on the following 6 sets.

SURVIVED	a numeric vector. Outcome, survival of democracy during the inter-war period.
DEVELOPED	a numeric vector. Condition, economically developed countries.
URBAN	a numeric vector. Condition, urbanized countries.
LITERATE	a numeric vector. Condition, countries with high literacy rate.
INDUSTRIAL	a numeric vector. Condition, industrialized countries.
STABLE	a numeric vector. Condition, politically stable countries.

Details

Data used by Ragin (2009) to illustrates the variants of QCA. Originally by Lipset (1959). Data are fuzzy-sets.

References

Lipset, S. M. (1959) "Some Social Requisites of Democracy: Economic Development and Political Legitimacy", American Political Science Review 53, pp. 69-105.

Ragin, C. C. (2009) "Qualitative Comparative Analysis. Using Fuzzy Sets (fsQCA)." In Rihoux, B., and Ragin, C. C. (eds.) Configurational Comparative Methods. Qualitative Comparative Analysis (QCA) and Related Techniques. Thousand Oaks, CA and London: Sage, pp. 87-121.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(LIPF)
```

`LIPR` *Lipset (1959), raw data*

Description

The `LIPR` data frame has 18 rows and 6 variables

Usage

```
data(LIPR)
```

Format

A data frame with 18 observations on the following 6 variables.

`SURVIVED` a numeric vector. Outcome, survival of democracy during the inter-war period.

`DEVELOPED` a numeric vector. Condition, level of economic development.

`URBAN` a numeric vector. Condition, level of urbanization.

`LITERATE` a numeric vector. Condition, level of literacy.

`INDUSTRIAL` a numeric vector. Condition, level of industrialization.

`UNSTABLE` a numeric vector. Condition, politically stable countries.

Details

Data used by Ragin (2009) to illustrates the variants of QCA. Originally by Lipset (1959). Data are raw-scores.

References

Lipset, S. M. (1959) "Some Social Requisites of Democracy: Economic Development and Political Legitimacy", American Political Science Review 53, pp. 69-105.

Ragin, C. C. (2009) "Qualitative Comparative Analysis. Using Fuzzy Sets (fsQCA)." In Rihoux, B., and Ragin, C. C. (eds.) Configurational Comparative Methods. Qualitative Comparative Analysis (QCA) and Related Techniques. Thousand Oaks, CA and London: Sage, pp. 87-121.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(LIPR)
```

`LR.intersect` *Function for identifying contradictory simplifying assumptions or easy counterfactuals.*

Description

A function for identifying contradictory simplifying assumptions or easy counterfactuals by intersecting the SAs or ECs of two solutions.

Usage

```
LR.intersect(results1, results2, sol1 = 1, sol2 = 1)
```

Arguments

`results1`	An object of class "qca". It can be a parsimonious or an intermediate solution for an outcome or its negation obtained via `minimize()`.
`results2`	An object of class "qca". It can be a parsimonious or an intermediate solution for an outcome or its negation obtained via `minimize()`.
`sol1`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.
`sol2`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.

Author(s)

Ioana-Elena Oana

See Also

minimize

Examples

```
# Import your data. For example:

data(SCHF)

# Get the parsimonious solution:

sol_yp <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE)

# Get the parsimonious solution for the absence of the outcome:

sol_nyp <- minimize(SCHF, outcome = "EXPORT", neg.out = TRUE,
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE)


# Get the contradictory simplofying assumptions:

LR.intersect(sol_yp, sol_nyp)
```

negateExp *Negates a boolean expression.*

Description

Function that negates a boolean expression.

Usage

```
negateExp(expression)
```

Arguments

`expression`	A boolean expression. Conditions should be capitalized and negated conditions should be inserted with a "~". Unions of conditions are performed with a "+", while intersections are performed with a "*".

Value

It returns a negated boolean expression.

Author(s)

Ioana-Elena Oana

Examples

```
negateExp("~EMP*MA")
negateExp("~A*B + C*~D")
```

PAYF *Paykani et al. (2018)*

Description

The PAYF data frame has 131 rows and 9 sets. The data is calibrated into fuzzy sets.

Usage

```
data(PAYF)
```

Format

A data frame with 131 observations on the following 9 sets.

COUNTRY	Country
REGION	Region the country belongs to.
HE	Condition: healthy education system
GG	Condition: good governance
AH	Condition: affluent health system
HI	Condition: high income inequality
HW	Condition: high wealth
HL	Outcome: high life expectancy
LL	Negated Outcome: low life expectancy

References

Paykani, Toktam, Rafiey, Hassan, and Sajjadi, Homeira. 2018. A fuzzy set qualitative comparative analysis of 131 countries: which configuration of the structural conditions can explain health better? International journal for equity in health, 17(1), 10.

Examples

```
data(PAYF)
```

PAYR *Paykani et al. (2018)*

Description

The `PAYR` data frame has 131 rows and 9 variables. The data is raw, uncalibrated.

Usage

```
data(PAYR)
```

Format

A data frame with 131 observations on the following 9 sets.

`COUNTRY` Country

`REGION` Region the country belongs to.

`LIFEX` Life expectancy, raw data.

`EDUC` Education, raw data.

`GOV` Governance, raw data.

`HEAL` Health system, raw data.

`INCEQ` Income inequality, raw data.

`WEAL` Wealth, raw data.

References

Paykani, Toktam, Rafiey, Hassan, and Sajjadi, Homeira. 2018. A fuzzy set qualitative comparative analysis of 131 countries: which configuration of the structural conditions can explain health better? International journal for equity in health, 17(1), 10.

Examples

```
data(PAYR)
```

PENF *Pennings (2003)*

Description

The PENF data frame has 45 rows and 5 sets

Usage

```
data(PENF)
```

Format

A data frame with 45 observations on the following 5 sets.

K a numeric vector. Outcome, constitutional control.

C a numeric vector. Condition, consensus democracy.

P a numeric vector. Condition, presidentialism.

N a numeric vector. Condition, new democracy.

R a numeric vector. Condition, rigid constritution.

Details

Data used by Pennings (2009) to explain constitutional control. Data are fuzzy-sets.

References

Pennings, P. (2009) "Beyond Dichotomous Explanations: Explaining Constitutional control of the Executive with Fuzzy-sets", European Journal of Political Research 42, pp. 541-567.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(PENF)
```

pimdata	*Function to extract prime implicants table from object of class "qca"*

Description

A function that displays each case's set membership scores in each sufficient term, the solution formula, and the outcome from an object of class "qca".

Usage

```
pimdata(results, outcome, sol = 1, ...)
```

Arguments

`results`	An object of class "qca". For performing pimdata of the sufficient solution for the negated outcome one must only use the `minimize()` result from the sufficiency analysis of the negated outcome.
`outcome`	A character string with the name of the outcome in capital letters. When performing pimdata of the sufficient solution for the negated outcome one must only use the `minimize()` result from the sufficiency analysis of the negated outcome in the argument `results`. Changing the name in the argument `outcome` or using a tilde is not necessary.
`sol`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.
`...`	Deprecated arguments (neg.out, use.tilde)

Value

A table with set memberships.

`solution_formula`	The solution formula.
`out`	Membership in the outcome.

Author(s)

Ioana-Elena Oana and Juraj Medzihorsky

See Also

`minimize` `pimplot`

Examples

```
# Import your data. For example:

data(SCHF)

# Get the parsimonious solution:


sol_yp <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE)

# Get the intermediate solution:

sol_yi <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE, dir.exp = c(0,0,0,0,0,0))


# Get the prime implicants table for the parsimonious solution:

pimdata(results = sol_yp, outcome = "EXPORT")

# Get the prime implicants table for the first intermediate solution:

pimdata(results = sol_yi, outcome = "EXPORT", sol = 1)
```

`pimplot` *Prime implicants, truth table rows, and necessity plots.*

Description

A function that displays XY plots for each sufficient term and the solution formula plotted against the outcome from an object of class "qca" (obtained by using the `minimize` function in package `QCA`). The function can also plot truth table rows against the outcome. Additionally, the function can plot results obtained from necessity analyses using an object of class "sS" (obtained by using the `superSubset` function in package `QCA`).

Usage

```
pimplot(data = NULL,
            results,
            outcome,
            incl.tt=NULL,
            ttrows= c(),
            necessity=FALSE,
            sol=1,
            all_labels=FALSE,
            markers = TRUE,
            labcol="black",
            jitter = FALSE,
            font = "sans",
            fontface = "italic",
            fontsize = 3,
            crisp = FALSE,
            consH = FALSE,
            ...)
```

Arguments

`data`	For analyses of sufficiency, providing a dataframe is not necessary. For analyses of necessity on objects of class "sS, you need to provide a dataframe with the name of the outcome and of the conditions in capital letters.
`results`	An object of class "qca" when necessity is `FALSE`. An object of class "sS" when necessity is `TRUE`. For performing pimplot of the sufficient solution for the negated outcome one must use the `minimize()` result from the sufficiency analysis of the negated outcome.
`outcome`	A character string with the name of the outcome in capital letters. When performing pimplot of the sufficient solution for the negated outcome one must only use the `minimize()` result from the sufficiency analysis of the negated outcome in the argument `results`. Changing the name in the argument `outcome` or using a tilde is not necessary, but recommended.
`incl.tt`	A numerical vector of length 1 specifying the row consistency threshold above which it should plot truth table rows. By default it is `NULL` and the function will produce plots using "qca" or "sS" objects. If a numerical value is specifyied, then it automatically only plots truth table rows above that consistency value. N.B. This argument cannot be used simultaneously with the `ttrows` argument.
`ttrows`	A vector of character strings specifying the names of the truth table rows to be printed. By default this vector is empty and the function will produce plots using "qca" or "sS" objects. If a value is specifyied, then it automatically only plots those particular truth table rows. N.B. This argument cannot be used simultaneously with the `incl.tt` argument.
`necessity`	logical. It indicates if the output should be for the results of sufficiency or necessity analyses. By default, `FALSE`, the function works with an object of class "qca" obtained from the `minimize` function in package `QCA`. When it set to `TRUE` the function returns plots for an object of class "sS" obtained from the `superSubset` function in package `QCA`.
`sol`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.

`all_labels`	Logical. Print ALL case labels?
`markers`	Logical. Print deviant consistency cases with different markers?
`labcol`	Color of the labels.
`jitter`	Logical. Should labels not overlab?
`font`	Font of the labels. Accepts "sans", "serif", and "mono" fonts.
`fontface`	Fontface of the labels. Accepts "plain", "bold", "italic", "bold.italic".
`fontsize`	Fontsize of the labels.
`crisp`	Logical. Should the function return a two-by-two table for crisp sets?
`consH`	Logical. Should Haesebrouck's consistency be printed?
`...`	Other non essential arguments.

Value

XY plots.

Author(s)

Ioana-Elena Oana

References

Haesebrouck, T. (2015) Pitfalls in QCA's consistency measure. Journal of Comparative Politics 2:65-80.

Schneider, C. Q., Rohlfing, I. 2013. Combining QCA and Process Tracing in Set-Theoretic Multi-Method Research. Sociological Methods Research 42(4): 559-597

See Also

minimize pimdata

Examples

```
# Import your data. For example:

data(SCHF)

# Get the parsimonious solution:


sol_yp <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE)

# Plot the prime implicants of the parsimonious solution:

pimplot(data = SCHF, results = sol_yp, outcome = "EXPORT")
```

```
# Plot a two-by-two table:

pimplot(data = SCHF, results = sol_yp, outcome = "EXPORT", crisp = TRUE)

# Plot all truth table rows with a consistency higher than 0.95:

pimplot(data=SCHF, results = sol_yp, incl.tt=0.97, outcome = "EXPORT", sol = 1)

# Plot truth table row "60":

pimplot(data=SCHF, results = sol_yp, ttrows =c("60"),
        outcome = "EXPORT", sol = 1)

# For plotting results of necessity analyses using superSubset,
# the first stept is to obtain an "sS" object:

SUPSUB <- superSubset(SCHF, outcome="EXPORT",
                      conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                      relation = "necessity", incl.cut = 0.996)
SUPSUB

# This can be imputed as result and necessity should be set to \code{TRUE}:

pimplot(data = SCHF, results = SUPSUB, outcome = "EXPORT", necessity = TRUE)
```

property.cube *Function producing a 3D scatter plot.*

Description

A function for visualizing 3D property spaces as a 3D scatter plot using the scatterplot3d package.

Usage

```
property.cube(data, labs = FALSE,
                             main = "3D Property Space",
                             xlab=NULL,
                             ylab=NULL,
                             zlab=NULL,
                             highlight.3d=TRUE,
                             dot.cex=0.5,
                             dot.col="black",
                             dot.srt=15,
                             dot.pos=3,
                             dot.offset = 1)
```

Arguments

`data`	A dataframe with 3 conditions.
`labs`	Logical. Should the case names be printed? If set to TRUE, it will automatically print the rownames of the dataframe given.
`main`	an overall title for the plot. The default is "3D Property Space"
`xlab`	a title for the x-axis. The default is the name of the first column in the dataframe.
`ylab`	a title for the y-axis. The default is the name of the third column in the dataframe.
`zlab`	a title for the z-axis. The default is the name of the second column in the dataframe.
`highlight.3d`	Logical. Should dots be colored differently according to their position in the property space?
`dot.cex`	size of the case labels
`dot.col`	color of the case labels
`dot.srt`	rotation of the case labels
`dot.pos`	position of the case labels (1-below, 2-left, 3-above, 4-right)
`dot.offset`	distance of text label from the dot

Value

It returns an enhanced 3d scatter plot using the `scatterplot3d` package.

Author(s)

Ioana-Elena Oana

Examples

```
# Load the Schneider data:

data(SCHF)

# Create a property space for conditions "EMP","BARGAIN", and outcome "EXPORT":

property.cube(SCHF[,c("EMP","BARGAIN","EXPORT")])

# Create a property space for conditions 1,2, and 3 in the data together with case labels:

property.cube(SCHF[,1:3], labs=TRUE)
```

`QCAfit` *Function calculating the parameters of fit*

Description

QCAfit is a function calculating parameters of fit useful in QCA and fsQCA that are consistency, coverage, PRI, Haesebrouck's consistency, RoN and PRODUCT. It works with both single and multiple conditions.

Usage

```
QCAfit(x, y, cond.lab = NULL, necessity = TRUE, neg.out = FALSE,
              product = FALSE, sol=1, ttrows= c(), consH = FALSE)
```

Arguments

`x`	A vector containing the values of a condition, a matrix with more than one conditions, or an object of class "qca" when necessity is `FALSE` and when outcome is specifyied as a character string.
`y`	A vector containing the values of the output or a character string when y is of class "qca".
`cond.lab`	When inserting a dataframe or a matrix with more than one condition and column names, the function automatically prints the names of the conditions tested. When inputing a vector, hence a single condition (i.e. a single column in a dataframe, the name of the condition tested should be inserted in this option .)
`necessity`	logical. It indicates if the output should be for sufficient or necessary condition(s). By default, `FALSE`, the function returns a table of parameters of fit for sufficient condition(s) (Consistency, Coverage, PRI, Haesebrouck's Consistency, and optionally Product). When it set to `TRUE` the function returns a table of parameters of fit for necessary condition(s) (Consistency, Coverage, Relevance of Necessity).
`neg.out`	logical. It indicates if the parameters of fit should be computed for the positive or the negative outcome. By default, `FALSE`, the function returns parameters of fit for the positive outcome .
`product`	logical. It indicates whether the parameter of fit PRODUCT should be shown. This stands for the product between the consistency sufficiency parameter and the PRI parameter.
`sol`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.
`ttrows`	A vector specifying the names of the truth table rows for which the function reports parameters of fit. For using this option y must be a "qca" object.
`consH`	Logical. Print also the Haesebrouck's consistency among the parameters of fit?

Value

It returns a matrix containing the parameters of fit for each condition.

Author(s)

Mario Quaranta and Ioana-Elena Oana

References

Haesebrouck, T. (2015) Pitfalls in QCA's consistency measure. Journal of Comparative Politics 2:65-80.

Ragin, C. C. 2006. Set Relations in Social Research: Evaluating Their Consistency and Coverage. Political Analysis 14(3): 291-310.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

See Also

minimize

Examples

```
# Generate fake data
set.seed(1234)

a <- runif(100, 0, 1)
b <- runif(100, 0, 1)
c <- runif(100, 0, 1)
y <- runif(100, 0, 1)

# Only one condition, for necessity
QCAfit(a, y, cond.lab = "A")

# With three conditions and their negation, for necessity
QCAfit(cbind(a, b, c), y)

# Only one condition, for sufficiency
QCAfit(a, y, cond.lab = "A", necessity = FALSE)

# With three conditions, their negation and negated output, for necessity
QCAfit(cbind(a, b, c), y, neg.out = TRUE)

# Load the Schneider data:

data(SCHF)

# Get parameters of fit for condition EMP as necessary for outcome EXPORT:

QCAfit(SCHF$EMP, SCHF$EXPORT, cond.lab = "EMP")

# Get parameters of fit for condition ~EMP as necessary for outcome ~EXPORT:
```

```
QCAfit(1-SCHF$EMP, SCHF$EXPORT, neg.out=TRUE, cond.lab = "~EMP")

# Get parameters of fit for all conditions and their negation as necessary for outcome EXPORT:

QCAfit(SCHF[,1:6], SCHF$EXPORT)

# Obtain the parsimonious solution for outcome "EXPORT":

sol_yp <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE)

# Get parameters of fit for the parsimonious solution:

QCAfit(x = sol_yp, y = "EXPORT", necessity = FALSE)

# Get parameters of fit for truth table rows 2,8, and 10:

QCAfit(x = sol_yp, y = "EXPORT", ttrows=c("2","8","10"), necessity = FALSE)
```

`QCAradar` *Function for displaying a radar chart.*

Description

Function displays a radar chart for an object of class "qca" or for a boolean expression.

Usage

```
QCAradar(results, outcome= NULL, fit= FALSE, sol = 1)
```

Arguments

`results`	An object of class "qca". For performing radar charts of the sufficient solution for the negated outcome one must only use the `minimize()` result from the sufficiency analysis of the negated outcome. The argument results can also be a boolean expression of the form e.g. "A*~B + ~B*C".
`outcome`	A character string with the name of the outcome in capital letters when results is of type 'qca'. When performing radar charts of the sufficient solution for the negated outcome one must only use the `minimize()`result from the sufficiency analysis of the negated outcome in the argument `results`. Changing the name in the argument `outcome` or using a tilde is not necessary.
`fit`	Logical. Print parameters of fit when results is of type 'qca'

`sol`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.

Author(s)

Ioana-Elena Oana

References

Maerz, F. Seraphine. 2017. "Pathways of Authoritarian Persistence." Paper presented at the CEU Annual Doctoral Conference

See Also

minimize

Examples

```
# Import data.
# For example:

data(SCHF)

# Get the intermediate solution:

sol_yi <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE, dir.exp = c(0,0,0,0,0,0))

# Display radar chart for the intermediate solution:

QCAradar(results = sol_yi, outcome = "EXPORT", fit=TRUE, sol = 1)

# Show a radar chart for the following boolean expression "A + ~B*Z*~C"

QCAradar(results = "A + ~B*Z*~C")
```

`rob.calibrange` *Function for identifying the calibration threshold ranges for a condition within which the Boolean formula for the solution does not change.*

Description

Function for identifying the calibration threshold ranges for a conditio within which the Boolean formula for the solution does not change. The function gradually increases and, then, decreases each of the three (0, 0,5, 1) calibration thresholds of a condition by the value specifyied in the step argument and checks whether the solution formula changes. The function performs this iteration for the number of times specified in the max.runs argument. If the solution formula does not change given the number of runs specified, it will return an NA, meaning that it could not find a limit to the range.

Usage

```
rob.calibrange(raw.data,
    calib.data,
    test.cond.raw,
    test.cond.calib,
    test.thresholds,
    type = "fuzzy",
    step = 1,
    max.runs = 20,
    outcome,
    conditions,
    incl.cut = 1,
    n.cut = 1,
    include = "",
    ...)
```

Arguments

`raw.data`	A data frame containing the raw data used to test the calibration thresholds for the chosen condition.
`calib.data`	A data frame containing the calibrated data for the sufficient solution.
`test.cond.raw`	A character string specifying the name of the condition in the RAW dataset for which to test the calibration ranges.
`test.cond.calib`	A character string specifying the name of the condition in the CALIBRATED dataset for which to test the calibration ranges.
`test.thresholds`	The initial qualitative anchors used for calibrating the chosen condition.
`type`	The calibration type, "fuzzy" is the default, but it can also be changed to "crisp".
`step`	The value to be gradually added and subtracted from the threshold tested.
`max.runs`	The maximum number of times the step value gets gradually added and subtracted.
`outcome`	A character string with the name of the outcome in capital letters. For the negated outcome a tilde "~" should be used. This had the same usage as the outcome argument in the minimize function.

`conditions`	A vector of character strings containing the names of the conditions.This had the same usage as the conditions argument in the minimize function.
`incl.cut`	The raw consistency threshold for the truth table rows.
`n.cut`	The frequency threshold for the truth table rows.
`include`	A vector of other output values (for example "?" for logical remainders) to include in the minimization. This had the same usage as the include argument in the minimize function.
`...`	Other options that the minimize function in the QCA package accepts. Check them out using ?minimize.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2020. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
## Not run:
# Load the calibrated data:
data(LIPF)

# Load the raw data:
data(LIPR)

# Get calibration ranges for condition DEVELOPED:

rob.calibrange(
  raw.data = LIPR,
  calib.data = LIPF,
  test.cond.raw = "DEVELOPED",
  test.cond.calib = "DEVELOPED",
  test.thresholds = c(400, 550, 900),
  step = 500,
  max.runs = 2,
  outcome  = "SURVIVED",
  conditions = c("URBAN","LITERATE","INDUSTRIAL","STABLE"),
  incl.cut = 0.8,
  n.cut = 1,
  include = "?"
)

## End(Not run)
```

rob.cases *Function for identifying cases in the intersections between an inital solution and test solutions.*

Description

Function for identifying cases between the various intersections between an inital solution and test solutions. The function also returns case ratio paramaters.

Usage

```
rob.cases(test_sol,
            initial_sol,
            outcome)
```

Arguments

`test_sol`	The different alternative solutions created with minimize() and placed in a list using list().
`initial_sol`	The initial solution created with minimize().
`outcome`	A character string containing the name of the outcome.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2020. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
## Not run:
# Load the data:
data(PAYF)

# Store the name of the conditions in one vector:
conds <-  c("HE","GG","AH","HI","HW")

# Create several solutions:

# The initial solution
IS <- minimize(data = PAYF,
                outcome  = "HL",
                conditions = conds,
                incl.cut = 0.87,
                n.cut = 2,
                include = "?",
                details = TRUE,
                show.cases = TRUE)

# altering consistency
TS1 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.7,
                 n.cut = 2,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

#altering n.cut
TS2 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.87,
                 n.cut = 1,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

# Create the test set in a list:
TS <- list(TS1, TS2)

# Looking at cases against the test set:

rob.cases(test_sol = TS,
                 initial_sol = IS,
                 outcome = "HL")
## End(Not run)
```

rob.corefit — *Function returning parameters of fit for the robust core of different alternative sufficient solutions.*

Description

Function returning parameters of fit for the robust core between an initial solution and various alternative test solutions.

Usage

```
rob.corefit(test_sol,
           initial_sol,
           outcome)
```

Arguments

`test_sol`	The different alternative solutions created with minimize() and placed in a list using list().
`initial_sol`	The initial solution created with minimize().
`outcome`	A character string containing the name of the outcome.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2020. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
# Load the data:
data(PAYF)

# Store the name of the conditions in one vector:
conds <-  c("HE","GG","AH","HI","HW")

# Create several solutions:

# The initial solution
IS <- minimize(data = PAYF,
                outcome  = "HL",
                conditions = conds,
                incl.cut = 0.87,
                n.cut = 2,
                include = "?",
                details = TRUE,
                show.cases = TRUE)

# altering consistency
TS1 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.7,
                 n.cut = 2,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

#altering n.cut
TS2 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.87,
                 n.cut = 1,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

# Create the test set in a list:
TS <- list(TS1, TS2)

# Calculate robustness parameters, i.e. the ratio of the parameters
# of fit for the core vis-a-vis for the initial solution:

CF <- rob.corefit(test_sol = TS,
           initial_sol = IS,
           outcome = "HL")
CF
```

`rob.fit` *Function returning robustness parameters of fit.*

Description

Function returning robustness parameters of fit of the intersections between an initial sufficient solution and various alternative test solutions.

Usage

```
rob.fit(test_sol,
               initial_sol,
               outcome)
```

Arguments

`test_sol`	The different alternative solutions created with minimize() and placed in a list using list().
`initial_sol`	The initial solution created with minimize().
`outcome`	A character string containing the name of the outcome.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2020. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
# Load the data:
data(PAYF)

# Store the name of the conditions in one vector:
conds <-  c("HE","GG","AH","HI","HW")

# Create several solutions:

# The initial solution
IS <- minimize(data = PAYF,
                outcome  = "HL",
                conditions = conds,
                incl.cut = 0.87,
                n.cut = 2,
                include = "?",
                details = TRUE,
                show.cases = TRUE)

# altering consistency
TS1 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.7,
                 n.cut = 2,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

#altering n.cut
TS2 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.87,
                 n.cut = 1,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

# Create the test set in a list:
TS <- list(TS1, TS2)

# Calculate robustness parameters, i.e. the ratio of the parameters
# of fit for the core vis-a-vis for the initial solution:

RF <- rob.fit(test_sol = TS,
           initial_sol = IS,
           outcome = "HL")
RF
```

`rob.inclrange` *Function for identifying the raw consistency threshold range within which the Boolean formula for the solution does not change.*

Description

Function for identifying the raw consistency threshold range for a truth table within which the Boolean formula for the solution does not change. The function gradually increases and, then, decreases an inital selected threshold by the value specifyied in the step argument and checks whether the solution formula changes for finding the lower and upper ranges for the raw consistency threshold. The function performs this iteration for the number of times specified in the max.runs argument. If the solution formula does not change given the number of runs specified, it will return an NA, meaning that it could not find a limit to the range.

Usage

```
rob.inclrange(data,
    step = 0.1,
    max.runs = 20,
    outcome,
    conditions,
    incl.cut = 1,
    n.cut = 1,
    include = "",
    ...)
```

Arguments

`data`	A data frame containing the calibrated data for the sufficient solution.
`step`	The value to be gradually added and subtracted from the threshold tested.
`max.runs`	The maximum number of times the step value gets gradually added and subtracted.
`outcome`	A character string with the name of the outcome in capital letters. For the negated outcome a tilde "~" should be used. This had the same usage as the outcome argument in the minimize function.
`conditions`	A vector of character strings containing the names of the conditions.This had the same usage as the conditions argument in the minimize function.
`incl.cut`	The raw consistency threshold for the truth table rows.
`n.cut`	The frequency threshold for the truth table rows.
`include`	A vector of other output values (for example "?" for logical remainders) to include in the minimization. This had the same usage as the include argument in the minimize function.
`...`	Other options that the minimize function in the QCA package accepts. Check them out using ?minimize.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2020. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
## Not run:
# Load the calibrated data:
data(PAYF)

# Check raw consistency ranges:

rob.inclrange(
  data = PAYF,
  step = 0.01,
  max.runs = 10,
  outcome  = "HL",
  conditions = c("HE","GG","AH","HI","HW"),
  incl.cut = 0.87,
  n.cut = 2,
  include = "?"
)

## End(Not run)
```

`rob.ncutrange` *Function for identifying the frequency threshold range within which the Boolean formula for the solution does not change.*

Description

Function for identifying the frequency threshold range for a truth table within which the Boolean formula for the solution does not change. The function gradually increases and, then, decreases an inital selected threshold by the value specifyied in the step argument and checks whether the solution formula changes for finding the lower and upper ranges for the frequency threshold. The function performs this iteration for the number of times specified in the max.runs argument. If the solution formula does not change given the number of runs specified, it will return an NA, meaning that it could not find a limit to the range.

Usage

```
rob.ncutrange(data,
    step = 1,
    max.runs = 20,
    outcome,
    conditions,
    incl.cut = 1,
    n.cut = 1,
    include = "",
    ...)
```

Arguments

`data`	A data frame containing the calibrated data for the sufficient solution.
`step`	The value to be gradually added and subtracted from the threshold tested.
`max.runs`	The maximum number of times the step value gets gradually added and subtracted.
`outcome`	A character string with the name of the outcome in capital letters. For the negated outcome a tilde "~" should be used. This had the same usage as the outcome argument in the minimize function.
`conditions`	A vector of character strings containing the names of the conditions.This had the same usage as the conditions argument in the minimize function.
`incl.cut`	The raw consistency threshold for the truth table rows.
`n.cut`	The frequency threshold for the truth table rows.
`include`	A vector of other output values (for example "?" for logical remainders) to include in the minimization. This had the same usage as the include argument in the minimize function.
`...`	Other options that the minimize function in the QCA package accepts. Check them out using ?minimize.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2020. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
# Load the calibrated data:
data(PAYF)

# Check frequency ranges:
rob.ncutrange(
  data = PAYF,
  step = 1,
  max.runs = 10,
  outcome  = "HL",
  conditions = c("HE","GG","AH","HI","HW"),
  incl.cut = 0.87,
  n.cut = 2,
  include = "?"
)
```

`rob.singletest` *Function returning robustness parameters for each single test solution in TS.*

Description

Function returning robustness parameters for each single test solution in TS. The function returns the set-coincidence and RC_Rank between each alternative solution and the initial solution.

Usage

```
rob.singletest(test_sol,
               initial_sol,
               outcome)
```

Arguments

`test_sol`	The different alternative solutions created with minimize() and placed in a list using list().
`initial_sol`	The initial solution created with minimize().
`outcome`	A character string containing the name of the outcome.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2021. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (2021). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
# Load the data:
data(PAYF)

# Store the name of the conditions in one vector:
conds <-  c("HE","GG","AH","HI","HW")

# Create several solutions:
```

```
# The initial solution
IS <- minimize(data = PAYF,
                outcome  = "HL",
                conditions = conds,
                incl.cut = 0.87,
                n.cut = 2,
                include = "?",
                details = TRUE,
                show.cases = TRUE)

# altering consistency
TS1 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.7,
                 n.cut = 2,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

#altering n.cut
TS2 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.87,
                 n.cut = 1,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

# Create the test set in a list:
TS <- list(TS1, TS2)

# Calculate robustness parameters, i.e. the ratio of the parameters
# of fit for the core vis-a-vis for the initial solution:

RST <- rob.singletest(test_sol = TS,
           initial_sol = IS,
           outcome = "HL")
RST
```

rob.xyplot *Function for plotting an initial solution against the test set.*

Description

Function for plotting an initial solution against the test set.

Usage

```
rob.xyplot(test_sol,
              initial_sol,
              outcome,
              all_labels = FALSE,
              jitter = TRUE,
              fontsize = 3,
              labs = TRUE,
              area_lab = TRUE)
```

Arguments

`test_sol`	The different alternative solutions created with minimize() and placed in a list using list().
`initial_sol`	The initial solution created with minimize().
`outcome`	A character string containing the name of the outcome.
`all_labels`	Logical. Should all the case labels be printed? If FALSE, only shaky and possible cases are printed.
`jitter`	Logical. Should case labels be jitter so as not to overlap?
`fontsize`	The size of the font for case labels.
`labs`	Logical.Should case labels be printed?
`area_lab`	Logical.Should labels for each area with regard to minTS/maxTS be printed?

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, and Carsten Q. Schneider. 2020. Robustness tests in QCA: A fit-oriented and case-oriented perspective using R. Unpublished Manuscript.

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
# Load the data:
data(PAYF)

# Store the name of the conditions in one vector:
conds <-  c("HE","GG","AH","HI","HW")

# Create several solutions:

# The initial solution
IS <- minimize(data = PAYF,
                outcome  = "HL",
                conditions = conds,
                incl.cut = 0.87,
                n.cut = 2,
                 include = "?",
                 details = TRUE,
                 show.cases = TRUE)

# altering consistency
TS1 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.7,
                 n.cut = 2,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

#altering n.cut
TS2 <- minimize(data = PAYF,
                 outcome  = "HL",
                 conditions = conds,
                 incl.cut = 0.87,
                 n.cut = 1,
                 include = "?",
                 details = TRUE, show.cases = TRUE)

# Create the test set in a list:
TS <- list(TS1, TS2)

# Plotting the initial solution against the test set:

rob.xyplot(test_sol = TS,
                  initial_sol = IS,
                  outcome = "HL",
                  fontsize = 2.5,
                  jitter=TRUE)
```

SAMF *Samford (2010)*

Description

The SAMF data frame has 61 rows and 4 sets

Usage

```
data(SAMF)
```

Format

A data frame with 61 observations on the following 4 sets.

Y a numeric vector. Outcome, trade liberalization.

G a numeric vector. Condition, lack of weak growth.

`H` a numeric vector. Condition, lack of hyper-inflation

`HorG` a numeric vector. Condition, H or G.

Details

Data are used by Samford (2010) to analyze rapid trade liberalization in Latin America. Data are fuzzy-sets.

References

Samford, S. (2010) "Averting 'Disruption and Reversal': Reassessing the Logic of Rapid Trade Reform in Latin America", Politics and Policy 38(3), pp. 373-407.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(SAMF)
```

`SC` *Selbst, practicing the truth table algorithm data*

Description

The `SC` data frame has 130 rows and 4 sets

Usage

```
data(SC)
```

Format

A data frame with 130 observations on the following 4 sets.

`A` a numeric vector. Condition, crisp-set.

`B` a numeric vector. Condition, crisp-set.

`C` a numeric vector. Condition, crisp-set.

`Y` a numeric vector. Condition, crisp-set.

Details

The authors of the data are Carsten Schnieder and Claudius Wagemann. The data are used in the on-line appendix of "Set-Theoretic Methods for the Social Sciences" to practice the truth table algorithm. Data are crisp-sets.

References

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Schneider, C. Q., Wagemann, C., Quaranta, M. (2012) How To... Use Software for Set-Theoretic Analysis. Online Appendix to "Set-Theoretic Methods for the Social Sciences". Available at www.cambridge.org/schneider-wagemann

Examples

```
data(SC)
```

SCHF *Schneider et. al. (2010)*

Description

The `SCHF` data frame has 76 observations and 7 sets

Usage

```
data(SCHF)
```

Format

A data frame with 76 observations on the following 7 sets.

`EMP` a numeric vector. Condition, employment protection.

`BARGAIN` a numeric vector. Condition, collective bargaining.

`UNI` a numeric vector. Condition, university training.

`OCCUP` a numeric vector. Condition, occupational training.

`STOCK` a numeric vector. Condition, stock market size.

`MA` a numeric vector. Condition, mergers and acquisitions.

`EXPORT` a numeric vector. Outcome, export performance in high-tech industries.

Details

Data used by Schneider et. al. (2010) to explain capitalist variety and export performance in high-tech industries. Data are fuzzy-sets.

References

Schneider, M. R., Schulze-Bentrop, C., Paunescu, M. (2010) "Mapping the institutional capital of high-tech firms: A fuzzy-set analysis of capitalist variety and export performance", Journal of International Business Studies 41, pp. 246-266.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(SCHF)
```

SCHLF *Schneider et. al. (2010)*

Description

The `SCHLF` data frame has 76 observations and 9 variables.

Usage

```
data(SCHLF)
```

Format

A data frame with 76 observations on the following 9 variables.

`EMP` a numeric vector. Condition, employment protection.

`BARGAIN` a numeric vector. Condition, collective bargaining.

`UNI` a numeric vector. Condition, university training.

`OCCUP` a numeric vector. Condition, occupational training.

`STOCK` a numeric vector. Condition, stock market size.

`MA` a numeric vector. Condition, mergers and acquisitions.

`EXPORT` a numeric vector. Outcome, export performance in high-tech industries.

`COUNTRY` a string vector. Name of the country in which the observation was made.

`YEAR` a numeric vector. Year in which the observation was made.

Details

Data used by Schneider et. al. (2010) to explain capitalist variety and export performance in high-tech industries. Data is saved in the long format and the first 7 variables are fuzzy-sets.

References

Schneider, M. R., Schulze-Bentrop, C., Paunescu, M. (2010) "Mapping the institutional capital of high-tech firms: A fuzzy-set analysis of capitalist variety and export performance", Journal of International Business Studies 41, pp. 246:266;

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(SCHLF)
```

SDC *Selbst, disappearing necessary condition data*

Description

The SDC data frame has 98 rows and 4 sets

Usage

```
data(SDC)
```

Format

A data frame with 98 observations on the following 4 sets.

A a numeric vector. Condition, crisp-set.

B a numeric vector. Condition, crisp-set.

C a numeric vector. Condition, crisp-set.

Y a numeric vector. Outcome, crisp-set.

Details

The authors of the data are Carsten Schnieder and Claudius Wagemann. The data are used in the on-line appendix of "Set-Theoretic Methods for the Social Sciences" to show the disappearance of a necessary condition. Data are crisp-sets.

References

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Schneider, C. Q., Wagemann, C., Quaranta, M. (2012) How To... Use Software for Set-Theoretic Analysis. Online Appendix to "Set-Theoretic Methods for the Social Sciences". Available at www.cambridge.org/schneider-wagemann

Examples

```
data(SDC)
```

skew.check *Function for checking how skewed sets are.*

Description

A function that identifies how skewed sets are by returning the number and percentage of cases with higher than 0.5 fuzzy-set values. The function can also return histograms of the calibrated sets.

Usage

```
skew.check(data, hist =  FALSE, main = NULL)
```

Arguments

`data`	A datafarme, a subset of a dataframe, or a vector (i.e. single column in a dataframe). The function should be used for calibrated data and will give an error if the data contains uncalibrated scores. However, if you have both calibrated and uncalibrated data in the same dataframe, it is possible to use the function only for the calibrated subset of that data.
`hist`	Logical. Should the function also return histograms of the sets?
`main`	Title for the plot. When provided with a dataframe, the function automatically assigns the column name to the plot.

Author(s)

Ioana-Elena Oana

Examples

```
# Import your data. For example:

data(SCHF)

# Check skewness for the entire dataframe:

skew.check(SCHF)

# Check skewness for the column "EMP" in the dataframe:

skew.check(SCHF$EMP)

# Check skewness for the 5th column of the dataframe:

skew.check(SCHF[,5])
```

`smmr`	*Function for performing set-theoretic multi-method research.*

Description

A function that selects best available cases for single case studies and best pairs of matching cases for comparative case studies.

Usage

```
smmr(results, outcome, sol = 1, match = NULL,
     cases = NULL, max_pairs = 5, term = 1, nec.cond =NULL, necessity = FALSE, ...)
```

Arguments

`results`	An object of class "qca".
`outcome`	A character string with the name of the outcome. If the negated outcome is analyzed, one can also write for example outcome = "~Y", however this is unnecessary as the values in the outcome will be taken from the appropriate minimized solution.
`sol`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.
`match`	Logical. Should comparative SMMR be used?

`cases`	A numerical vector indicating the type of cases to be returned.
	For single case studies for SUFFICIENCY:
	`1 = typical cases,`
	`2 = typical cases for each focal conjunct in a sufficient term,`
	`3 = deviant consistency,`
	`4 = deviant coverage,`
	`5 = individually irrelevant,`
	`6 = all of the above;`
	For comparative case studies for SUFFICIENCY:
	`1 = Typical-Typical for each focal conjuct in a sufficient term,`
	`2 = Typical-IIR for each focal conjuct in a sufficient term,`
	`3 = Typical-Dev.Cons.,`
	`4 = Dev.Cov.-IIR;`
	`5 = Typical-Typical for each term;`
	`6 = Typical-IIR for each term;`
	`7 = all of the above;`
	For single case studies for NECESSITY:
	`1 = typical cases,`
	`2 = deviant consistency,`
	`3 = deviant relevance,`

	`4 = all of the above;`
	For comparative case studies for NECESSITY:
	`1 = Typical-Deviant Relevance ,`
	`2 = Typical-Deviant Consistency,`
	`3 = Typical-IIR,`
	`4 = Typical-Typical,`
	`5 = all of the above;`

`max_pairs`	Maximum number of pairs to extract.
`term`	A numeric vector where the first number indicates the number of the term according to the order in the "qca" object.
`nec.cond`	A character vector specifying the necessity results obtained. This could be of the form "A", "A+B", or "A*B", making sure no spaces are left between the name of the conditions and the Boolean operator. Notice than nec.cond can also be specified for sufficiency results in single SMMR typical cases for each focal conjunct in a sufficient term, and for comparative SMMR in typical-typical and typical-iir comparisons for each focal conjunct. Note that in the later case only expressions of the forms "A" or "A+B" are accepted. If dealing with a necessary condition of the form "A*B", each of the individual conditions should be entered separately when performing analyses of sufficiency.
`necessity`	Logical. Should SMMR results for necessity be shown?
`...`	Deprecated arguments (neg.out, use.tilde)

Author(s)

Ioana-Elena Oana

References

Schneider, C. Q., Rohlfing, I. 2013. Combining QCA and Process Tracing in Set-Theoretic Multi-Method Research. Sociological Methods and Research 42(4): 559-97

See Also

`minimize`

Examples

```
# Import your data. For example:

data(SCHF)

## SUFFICIENCY SMMR examples:

# Get the parsimonious solution:

sol_yp <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE)

# Get typical cases for each focal conjunct in the third term of the parsimonious solution:

smmr(results = sol_yp, outcome = "EXPORT", match=FALSE, cases=2, term = 3)

# Get typical cases for each focal conjunct in the third term of the parsimonious solution
# and specifying STOCK as a necessary condition:

smmr(results = sol_yp, outcome = "EXPORT",
      match=FALSE, cases=2, term = 3, nec.cond="STOCK")

# Get matching typical-typical cases for the second term of the parsimonious solution:

smmr(results = sol_yp, outcome = "EXPORT", match=TRUE, cases=1, term = 2)

# Get matching typical-DCN cases:

smmr(results = sol_yp, outcome = "EXPORT", match=TRUE, cases=3)

## NECESSITY SMMR examples:

# Imagine you found condition "STOCK + MA" is nececssary for outcome "EXPORT".

# Get typical cases for each disjunct of the necessary condition:

smmr(results = sol_yp, outcome = "EXPORT", match=FALSE, cases=1,
     nec.cond = "STOCK+MA", necessity = TRUE)

# Get typical - deviant relevance cases for each disjunct of the necessary condition:

smmr(results = sol_yp, outcome = "EXPORT", match=TRUE, cases=1,
     nec.cond = "STOCK+MA", necessity = TRUE)
```

`stargazerSol` *Function for exporting a sufficienct solution from minimize in latex, html, or text format.*

Description

Function for exporting a sufficienct solution from minimize in latex, html, or text format.

Usage

```
stargazerSol(results,
            outcome,
            sol = 1,
            show.cases = FALSE,
            type = "latex",
            title = "",
            out = NULL,
            digits = 3)
```

Arguments

results	An object of class "qca": a sufficient solution obtained with the minimize function.
outcome	A character string with the name of the outcome in capital letters. When performing pimplot of the sufficient solution for the negated outcome one must only use the minimize() result from the sufficiency analysis of the negated outcome in the argument results. Changing the name in the argument outcome or using a tilde is not necessary, but recommended.
sol	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.
show.cases	Logical. Should the names of cases be printed?
type	character string that specifies what type of output the command should produce. The possible values are "latex" (default), "html", "text".
title	title for the table.
out	name of the file to be saved containing the extension (e.g. "mysol.tex", "mysol.txt")
digits	To how many digits should the parameters of fit be rounded up.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
# Import your data. For example:

data(SCHF)

# Get the parsimonious solution:


sol_yp <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE)

# Print in latex format:

stargazerSol(sol_yp, "EXPORT")
```

`stargazerTT` *Function for exporting a sufficienct solution from minimize in latex, html, or text format.*

Description

Function for exporting a sufficienct solution from minimize in latex, html, or text format.

Usage

```
stargazerTT(truthtable,
            show.cases = FALSE,
            type = "latex",
            title = "",
            out = NULL,
            digits = 3)
```

Arguments

`truthtable`	A truth table obtained with the truthTable function.
`show.cases`	Logical. Should the names of cases be printed?
`type`	character string that specifies what type of output the command should produce. The possible values are "latex" (default), "html", "text".
`title`	title for the table.
`out`	name of the file to be saved containing the extension (e.g. "mysol.tex", "mysol.txt")
`digits`	To how many digits should the parameters of fit be rounded up.

Author(s)

Ioana-Elena Oana

References

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
# Import your data. For example:

data(SCHF)

# Get the truth table:

mytt <- truthTable(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9, complete = TRUE)

# Export as latex:
stargazerTT(mytt)

# Export as text:
stargazerTT(mytt, type = "text")
```

`STUF` *Fake fuzzy-set student data.*

Description

Fake fuzzy-set football practice data used for Chapter 4 of the Oana et al. (forthcoming) book.

Usage

```
data(STUF)
```

Format

A data frame with 17 observations on the following 3 variables. Conditions for arriving in time for football practice.

`A`	Condition: Live close by
`B`	Condition: Arrive by bike
`C`	Condition: Come from punctual familie
`D`	Condition: Are seasoned players
`E`	Condition: Goalkeepers
`Y`	Outcome: Arrive on time for football practice

References

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
data(STUF)
```

`STUR` *Fake raw, student data.*

Description

Fake raw, student data used for Chapter 2 of the Oana et al. (forthcoming) book.

Usage

```
data(STUR)
```

Format

A data frame with 17 observations on the following 3 variables.

`MARK` Cond.: student mark

`PARTICIPATION` Cond.: quality and correctness of the student's comments in class,

`PEERS` Cond.: extent to which the student engages with and helps peers during interactive in-class teaching exercises

References

Oana, Ioana-Elena, Carsten Q. Schneider, and Eva Thomann (forthcoming). Qualitative Comparative Analysis (QCA) using R: A Gentle Introduction. Cambridge: Cambridge University Press.

Examples

```
data(STUR)
```

`theory.evaluation` *Performs theory evaluation.*

Description

Function that returns membership of cases in the intersections between theory and the empirical solution in the form of a data frame, the names of cases in the intersections between theory and the empirical solution, and the parameters of fit for these intersections.

Usage

```
theory.evaluation(theory, empirics, outcome, sol = 1, print.fit=FALSE,
                  print.data=FALSE, consH = FALSE, ...)
```

Arguments

`theory`	A character string specifying the theory. Unions of conditions are performed with a "+", while intersections are performed with a "*". Conditions should be capitalized and negated conditions should be inserted with a "~".
`empirics`	An object of class 'qca'. When performing analyses for the negated outcome, just use the results from the `minimize()` function for the negation of the outcome.
`outcome`	A character string with the name of the outcome. When performing analyses of the sufficient solution for the negated outcome one must only use the `minimize()` result from the sufficiency analysis of the negated outcome in the argument `empirics`. Changing the name in the argument `outcome` or using a tilde is not necessary.
`sol`	A vector where the first number indicates the number of the conservative or parsimonious solution according to the order in the "qca" object. For more complicated structures of model ambiguity, the intermediate solution can also be specified by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.
`print.data`	Logical. Print also the membership of cases in all the intersections between theory and empirics?
`print.fit`	Logical. Print also the parameters of fit for the intersections between theory and empirics?
`consH`	Logical. Print also the Haesebrouck's consistency among the parameters of fit?
`...`	Deprecated arguments (use.tilde).

Author(s)

Ioana-Elena Oana

References

Ragin, C. C. 1987. The Comparative Method: Moving Beyond Qualitative and Quantitative Strategies. Berkeley: University of California Press, pp. 118-121

Schneider, C. Q., Wagemann, C. 2012. Set-Theoretic Methods for the Social Sciences: A Guide to Qualitative Comparative Analysis. Cambridge: Cambridge University Press, chapter 11.3

See Also

`minimize`

Examples

```
## Not run:
# Import your data. For example:

data(SCHF)

# Get the intermediate solution:
```

```
sol_yi <- minimize(SCHF, outcome = "EXPORT",
                conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                incl.cut = .9,
                include = "?",
                details = TRUE, show.cases = TRUE, dir.exp = c(0,0,0,0,0,0))


# Specify the theory. Let's assume the theory says that the
# absence of EMP and the presence of MA is sufficient for EXPORT:

t<-"~EMP*MA"


# Perform theory evaluation (get only the names of the cases and the Boolean intersections):

TH <- theory.evaluation(theory = t, empirics = sol_yi, outcome = "EXPORT", sol = 1,
                  print.data=FALSE, print.fit=FALSE)
TH

# Get only the case names:

TH$cases

# Or only the parameters of fit:

TH$fit
## End(Not run)
```

THOF *Thomann (2015)*

Description

The THOF data frame has 76 rows and 7 sets. The study analyses how European Union (EU) member states adapt EU directives to domestic contexts during transposition and asks how and why fully compliant countries 'customize' EU directives.

Usage

```
data(THOF)
```

Format

A data frame with 76 observations on the following 7 sets.

RESP	Cond.: Responsive regulatory mode
SAL	Cond.: Salient issue

`RES` Cond.: Domestic resistance

`VPO` Cond.: Many veto points

`VPL` Cond.: Many veto players

`COERC` Cond.: Coercive interventionist style

`CUSTOM` Outcome: Extensive customization

Details

Data used by Thomann (2015) to explain customization. Data are fuzzy-sets.

References

Thomann, E. 2015. Customizing Europe: Transposition as bottom-up implementation. Journal of European Public Policy 22(10): 1368-1387.

Examples

```
data(THOF)
```

`VISC` *Vis (2009), crisp set data*

Description

The `VISC` data frame has 25 rows and 4 sets

Usage

```
data(VISC)
```

Format

A data frame with 25 observations on the following 4 sets.

`P` a numeric vector. Condition, weak political positions, with parties in government expecting losses at the next election.

`S` a numeric vector. Condition, deteriorating economic situation.

`R` a numeric vector. Condition, government dominated by parties from the right of the political spectrum.

`U` a numeric vector. Outcome, unpopular reform.

Details

Data are used by Vis (2009) to analyze the pursuit of unpopular reforms by governments. Data are crisp-sets.

References

Vis, B. (2009) "Government and Unpopular Social Policy Reforms: Biting the Bullet or Steering Clear?", European Journal of Political Research 48, pp. 31-57.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(VISC)
```

`VISF` *Vis (2009), fuzzy set data*

Description

The `VISF` data frame has 25 rows and 4 sets

Usage

```
data(VISF)
```

Format

A data frame with 25 observations on the following 4 sets.

- `p` a numeric vector. Condition, weak political positions, with parties in government expecting losses at the next election.
- `s` a numeric vector. Condition, deteriorating economic situation.
- `r` a numeric vector. Condition, government dominated by parties from the right of the political spectrum.
- `u` a numeric vector. Outcome, unpopular reform.

Details

Data are used by Vis (2009) to analyze the pursuit of unpopular reforms by governments. Data are fuzzy-sets.

References

Vis, B. (2009) "Government and Unpopular Social Policy Reforms: Biting the Bullet or Steering Clear?", European Journal of Political Research 48, pp. 31-57.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge University Press: Cambridge.

Examples

```
data(VISF)
```

xy.plot *Function producing enhanced XY plots*

Description

xy.plot produces XY plots and provides values for consistency, Haesebrouck's consistency, coverage, RoN, PRI. Several graphic parameters can be decided by the user.

Usage

```
xy.plot(x, y, data,
           labcol = "black",
           main = "XY plot",
           ylab = "Outcome",
           xlab = "Condition",
           necessity = FALSE,
           jitter = FALSE,
           font = "sans",
           fontface = "italic",
           fontsize = 3,
           labs = rownames(data),
           crisp = FALSE,
           shape = 19,
           consH = FALSE,
           ...)
```

Arguments

`x`	vector containing the condition.
`y`	vector containing the outcome.
`data`	The dataset used
`labcol`	color of the dots.
`main`	an overall title for the plot. The default is `"XY plot"`. See ?title.
`ylab`	a title for the y-axis. The default is `"Outcome"`. See ?title.
`xlab`	a title for the x-axis. The default is `"Condition"`. See ?title.
`necessity`	logical. Indicates if the parameters of fit are calculated for a sufficient or necessary condition. The default is `FALSE`, therefore it calculates the parameters of fit for sufficiency. To get the parameters of fit for necessary conditions set `necessity` as `TRUE`.
`jitter`	Logical. Should labels be jitter to not overlap?
`font`	Font of the labels. Accepts "sans", "serif", and "mono" fonts.
`fontface`	Fontface of the labels. Accepts "plain", "bold", "italic", "bold.italic".
`fontsize`	Fontsize of the labels.

`labs`	the vector of case labels. The default is the rownames of the dataset.
`crisp`	Logical. Should a two-by-two table for crisp sets be returned?
`shape`	The shape for the markers.
`consH`	Logical. Should Haesebrouck's consistency be printed?
`...`	Other internal arguments. Do not specify!

Value

It returns an enhanced XY plot.

Author(s)

Mario Quaranta and Ioana-Elena Oana.

References

Haesebrouck, T. (2015) Pitfalls in QCA's consistency measure. Journal of Comparative Politics 2:65-80.

Ragin, C. C. (2008) Redesigning Social Inquiry: Fuzzy Sets and Beyond. The Chicago University Press: Chicago and London.

Schneider, C. Q., Wagemann, C. (2012) Set-Theoretic Methods for the Social Sciences, Cambridge Univeristy Press: Cambridge.

Examples

```
# Load the Schneider data:

data(SCHF)

# Plot of condition EMP as necessary for outcome EXPORT with case labels
# and names for the plot and axes:

xy.plot("EMP", "EXPORT", data=SCHF, necessity = TRUE,
          main = "EMP as necessary for EXPORT", ylab = "EXPORT", xlab = "EMP")
```

Index