

Package ‘QCA’

April 25, 2012

Type Package

Title Qualitative Comparative Analysis

Version 1.0-3

Date 2012-04-24

Author Adrian Dusa <dusadrian@unibuc.ro> and Alrik Thiem <thiem@sipo.gess.ethz.ch>

Maintainer Adrian Dusa <dusadrian@unibuc.ro>

Depends R (>= 2.1.10), lpSolve

Suggests XML

Description Performs the Quine-McCluskey algorithm for Qualitative Comparative Analysis, for all versions: csQCA, mvQCA and fsQCA.

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-04-25 11:06:52

R topics documented:

QCA-package	2
allExpressions	3
calibrate	4
createMatrix	7
demoChart	8
Emme	9
eqmcc	10
factorize	14
findSubsets	15
findSupersets	16
findTh	18
getRow	19

HarKem	20
Krook	20
pof	21
RagStr	23
readTosmana	24
Rokkan	24
solveChart	25
superSubset	26
truthTable	28

Index	32
--------------	-----------

QCA-package

Qualitative Comparative Analysis

Description

This package implements the method of Qualitative Comparative Analysis (QCA) as developed by Ragin (1987, 2000, 2008). QCA is a configurational comparative technique which uses Boolean instead of linear algebra. This difference changes the understanding of causation as symmetric (linear algebra) to an understanding of causation as asymmetric (Boolean algebra).

QCA is based on set relations between a number of (explaining) conditions and an (explained) outcome, with the two concepts of necessity and sufficiency at its core. If some condition or combination of conditions is a superset of the outcome, the former is necessary for the latter. If some condition or combination of conditions is a subset of the outcome, the former is sufficient for the latter.

Three related variants of QCA exist: crisp-set QCA (csQCA), multi-value QCA (mvQCA) and fuzzy-set QCA (fsQCA). A subvariant of csQCA called temporal QCA (tQCA) is suitable for event sequence relations.

In this package, the "exact" Quine-McCluskey algorithm for crisp sets is implemented for csQCA and mvQCA (from version 0.6-0), and fsQCA as well as tQCA from version 1.0-0. Starting with version 0.4-5 the package has a new function called "eqmcc" ("e"nhanced Quine-McCluskey) which finds exact solutions faster and with substantially lower memory consumption (Dusa 2007, 2010).

How to make the most of R's and the QCA package's capabilities for QCA is demonstrated in detail with many examples by Thiem and Dusa (2012).

Details

Package: QCA
 Type: Package
 Version: 1.0-3
 Date: 2012-04-24
 License: GPL (>= 2)

Author(s)**Authors:**

Adrian Dusa
Department of Sociology
University of Bucharest
<dusadrrian@unibuc.ro>

Alrik Thiem
Department of Humanities, Social and Political Sciences
Swiss Federal Institute of Technology Zurich
<thiem@sipo.gess.ethz.ch>

Maintainer:

Adrian Dusa

References

- A. Dusa. *Enhancing Quine-McCluskey*. WP 2007-49, COMPASSS, 2007.
URL: <http://www.compasss.org/files/WPfiles/Dusa2007a.pdf>.
- A. Dusa. A Mathematical Approach to the Boolean Minimization Problem. *Quality & Quantity*, 44(1): 99-113, 2010.
- C. C. Ragin. *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987.
- C. C. Ragin. *Fuzzy-Set Social Science*, University of Chicago Press, Chicago, 2000.
- C. C. Ragin. *Redesigning Social Inquiry: Fuzzy Sets and Beyond*. University of Chicago Press, Chicago, 2008.
- A. Thiem and A. Dusa. *Qualitative Comparative Analysis with R: A User's Guide*. Springer, New York, 2012.

allExpressions

Create a Matrix with all Possible Combinations of Conditions

Description

There are exactly 3^k possible expressions (where k is the number of conditions) for a binary crisp-set procedure. For multi-value crisp data, the general formula for the number of possible expressions is $\text{prod}(\text{noflevels} + 1)$, where `noflevels` is a vector with the number of levels for each causal condition. As the matrix grows exponentially, it is not recommended to run this function for a large number of conditions (> 15).

Usage

```
allExpressions(noflevels, raw = FALSE, arrange = FALSE)
```

Arguments

noflevels	The number of levels for each causal condition
raw	Logical, return the raw result matrix; if FALSE, print the result matrix without NAs
arrange	Logical, try to arrange the result matrix for visual inspection (takes a lot of additional time for large matrices)

Value

A matrix with `prod(noflevels + 1)` rows

References

C. C. Ragin. *Fuzzy-Set Social Science*, University of Chicago Press, Chicago, 2000.

See Also

[createMatrix](#)

Examples

```
# for 3 conditions
allExpressions(rep(3, 3))

# the same matrix, this time arranged better
allExpressions(rep(3, 3), arrange = TRUE)
```

calibrate

Calibrate Crisp (Binary or Multi-Value) and Fuzzy Sets

Description

This function produces crisp and fuzzy sets from *raw data* (base variables) and some specified threshold(s). The calibration of fuzzy sets is possible for end-point and mid-point concepts, using the method of transformational assignment.

Usage

```
calibrate(x, type = "crisp", thresholds = NA, include = TRUE, logistic = FALSE,
          idm = 0.95, ecdf = FALSE, p = 1, q = 1)
```

Arguments

<code>x</code>	An interval or ratio-scaled base variable
<code>type</code>	The calibration type, either "crisp" or "fuzzy"
<code>thresholds</code>	A vector of thresholds
<code>include</code>	Logical, include threshold(s) in the set (<code>type = "crisp"</code> only)
<code>logistic</code>	Calibrate to fuzzy sets using the logistic function
<code>idm</code>	The set inclusion degree of membership for the logistic function
<code>ecdf</code>	Calibrate to fuzzy sets using the empirical cumulative distribution function (ECDF) of the base variable
<code>p</code>	Parameter: if $p > 1$ concentration, if $0 < p < 1$ dilation below crossover
<code>q</code>	Parameter: if $q > 1$ dilation, if $0 < q < 1$ concentration above crossover

Details

Calibration is the process by which set membership scores are assigned to cases. With interval and ratio-scaled base variables, calibration can be based on transformational assignments using (piecewise-defined) membership functions.

For `type = "crisp"`, one threshold produces crisp sets with two groups - 0s and 1s. More thresholds produce multi-value data. For example, two thresholds produce three groups - 0s, 1s and 2s.

For `type = "fuzzy"`, this function can generate linear, s-shaped and inverted s-shaped fuzzy numbers with respect to end-point concepts, including logistic transformations. It can generate trapezoidal, triangular and bell-shaped fuzzy numbers with respect to mid-point concepts (Bojadziew and Bojadziew, 2007; Clark et al., 2008; Thiem and Dusa, 2012).

For calibrating fuzzy sets based on end-point concepts, thresholds should be specified as a numeric vector `thresholds = c(thEX, thCR, thIN)`, where `thEX` is the threshold for full set exclusion, `thCR` the threshold for set crossover, and `thIN` the threshold for full set inclusion.

If $thEX < thCR < thIN$, then the membership function is increasing from `thEX` to `thIN`.

If $thIN < thCR < thEX$, then the membership function is decreasing from `thIN` to `thEX`.

For calibrating fuzzy sets based on mid-point concepts, thresholds should be specified as a vector `thresholds = c(thEX1, thCR1, thIN1, thIN2, thCR2, thEX2)`, where `thEX1` is the first (left) threshold for full set exclusion, `thCR1` the first (left) threshold for set crossover, `thIN1` the first (left) threshold for full set inclusion, `thIN2` the second (right) threshold for full set inclusion, `thCR2` the second (right) threshold for set crossover, and `thEX2` the second (right) threshold for full set exclusion.

If $thEX1 < thCR1 < thIN1 \leq thIN2 < thCR2 < thEX2$, then the membership function is first increasing from `thEX1` to `thIN1`, then decreasing from `thIN2` to `thEX2`.

If $thIN1 < thCR1 < thEX1 \leq thEX2 < thCR2 < thIN2$, then the membership function is first decreasing from `thIN1` to `thEX1`, then increasing from `thEX2` to `thIN2`.

The parameters `p` and `q` control the degree of concentration and dilation. They should be left at their default values unless good reasons to change them exist.

If `logistic = TRUE`, the argument `idm` specifies the inclusion degree of membership in the set. The exclusion degree is automatically determined because of the symmetry of the logistic function.

If `ecdf = TRUE`, calibration is based on the empirical cumulative distribution function of x . The arguments `logistic` and `ecdf` are mutually exclusive, with the first taking precedence over the second.

Value

A numeric vector of crisp set values or fuzzy set membership scores.

References

G. Bojadziev and M. Bojadziev. *Fuzzy Logic for Business, Finance, and Management*. 2nd ed., World Scientific, New Jersey, 2007.

Clark, T. D., J. M. Larson, J. N. Mordeson, J. D. Potter, and M. J. Wierman. *Applying Fuzzy Mathematics to Formal Models in Comparative Politics*. Springer, Berlin, 2008.

A. Thiem and A. Dusa. *Qualitative Comparative Analysis with R: A User's Guide*. Springer, New York, 2012.

Examples

```
# base variable; random draw from standard normal distribution
x <- rnorm(30)

# calibration thresholds
th <- quantile(x, seq(from = 0.1, to = 0.9, length = 5))

# calibration of binary-value crisp sets (two groups)
calibrate(x, thresholds = th[3])

# calibration of multi-value crisp set (three groups)
calibrate(x, thresholds = c(th[2], th[4]))

# fuzzy-set calibration (positive end-point concept, linear)
plot(x, calibrate(x, type = "fuzzy", thresholds = c(th[1], th[3], th[5])),
     ylab = "Fuzzy Set Membership")

# fuzzy-set calibration (positive end-point concept, logistic)
plot(x, calibrate(x, type = "fuzzy", thresholds = c(th[1], th[3], th[5]),
     logistic = TRUE, idm = 0.99), ylab = "Fuzzy Set Membership")

# fuzzy-set calibration (positive end-point concept, ECDF)
plot(x, calibrate(x, type = "fuzzy", thresholds = c(th[1], th[3], th[5]),
     ecdf = TRUE), ylab = "Fuzzy Set Membership")

# fuzzy-set calibration (negative end-point concept, s-shaped)
plot(x, calibrate(x, type = "fuzzy", thresholds = c(th[5], th[3], th[1]),
     p = 2, q = 2), ylab = "Fuzzy Set Membership")

# fuzzy-set calibration (positive mid-point concept, triangular)
plot(x, calibrate(x, type = "fuzzy", thresholds = th[c(1,2,3,3,4,5)]),
     ylab = "Fuzzy Set Membership")
```

```
# fuzzy-set calibration (negative mid-point concept, bell-shaped)
plot(x, calibrate(x, type = "fuzzy", thresholds = th[c(3,2,1,5,4,3)],
  p = 3, q = 3), ylab = "Fuzzy Set Membership")
```

createMatrix *Create the prod(k) matrix*

Description

The $\text{prod}(k + 1)$ matrix is the generalisation of the 3^k matrix, for any combination of binary and multi-value crisp sets, where k is a vector with the number of levels for each causal condition.

For example, the truth table consists of all combinations of the presence and absence of conditions (coded binary 1/0). There are 2^k such combinations and `createMatrix()` should be among the fastest to create this matrix, optimising ideas inspired from the base R function `expand.grid()`.

The $\text{prod}(k + 1)$ matrix has exactly the same structure as the 3^k matrix, given that, for instance, $3^3 = 3*3*3$, which is simply the same as `prod(rep(3, 3))`. It is easy to observe that 3^k is just a special case of `prod(rep(3, k))`.

For multi-values, the structure 3^k cannot hold, as each causal combination can have different number of levels. This issue is easily solved by using `createMatrix()` instead.

Usage

```
createMatrix(noflevels, logical = FALSE)
```

Arguments

<code>noflevels</code>	A vector containing the number of levels for each variable in the dataset
<code>logical</code>	Logical, return the matrix in logical values (only binary-value crisp data)

References

A. Dusa. A Mathematical Approach to the Boolean Minimization Problem. *Quality & Quantity*, 44(1), pp.99-113, 2010.

A. Dusa. *Enhancing Quine-McCluskey*. WP 2007-49, COMPASSS, 2007.
URL: <http://www.compasss.org/files/WPfiles/Dusa2007a.pdf>.

C. C. Ragin. *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987.

See Also

[allExpressions](#) [truthTable](#)

Examples

```
# create a binary 2^k matrix based on 3 conditions, essentially the truth table
noflevels <- rep(2, 3)
createMatrix(noflevels)

# its 3^k matrix, where all the prime implicants are, where "-1" means a minimised literal
createMatrix(noflevels + 1) - 1

# this is essentially the same structure as the more familiar
# (N.B. there is also a "raw" argument)
allExpressions(noflevels)

# or even more familiar, with all possible PIs arranged
allExpressions(noflevels, arrange = TRUE)

# create a truth table matrix based on 3 conditions where the second has three levels
# the matrix has 2*3*2 = prod(c(2, 3, 2)) = 12 rows
noflevels <- c(2, 3, 2)
createMatrix(noflevels)

# its prod(k) matrix, the generalised version of the 3^k matrix
# which has prod(noflevels + 1) rows
createMatrix(noflevels + 1) - 1
```

demoChart

Create the Prime Implicants Chart

Description

This function creates a chart having the prime implicants on the rows and the observed combinations of conditions on the columns. It is useful to determine visually which prime implicant (if any) is essential. The chart is subsequently processed algorithmically to further reduce the redundant prime implicants. This function is for demonstration purposes only.

Usage

```
demoChart(rows, columns, splitmethod = "")
```

Arguments

rows	A vector of strings, containing the prime implicants
columns	A vector of strings, containing all combinations of conditions from the original data
splitmethod	String, to declare the separator of the input strings

Value

a logical matrix showing which conditions from the (minimized) prime implicants are found in which columns

References

W.V. Quine. The Problem of Simplifying Truth Functions. *American Mathematical Monthly*, 59(8):521-531, 1952.

C. C. Ragin. *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987.

See Also

[prettyTable](#)

Examples

```
chart <- demoChart(c("A", "B", "c"), c("ABC", "Abc", "AbC", "aBc"))
prettyTable(chart)
```

```
# Quine's example, page 528
rows <- c("AB", "BC", "Ac", "aC", "abd", "bcd")
cols <- c("ABCD", "ABCd", "ABcD", "ABcd", "AbcD", "Abcd",
         "aBCD", "aBCd", "abCD", "abCd", "abcd")
```

```
chart <- demoChart(rows, cols)
prettyTable(chart)
```

 Emme

Job Security Regulations in Western Democracies

Description

The data frame **Emme** has 19 rows and 7 columns.

This data is from the study by Emmenegger (2011), which analyzes the determinants of high job security regulations in Western democracies using fsQCA.

Usage

```
data(Emme)
```

Format

The data contains the following sets:

- S** Condition: high level of statism
- C** Condition: high level of non-market coordination
- L** Condition: high level of labour movement strength
- R** Condition: high level of Catholicism
- P** Condition: high level of religious party strength
- V** Condition: many institutional veto points
- JSR** Outcome : high level of job security regulations

Note

The row names are the official International Organization for Standardization (ISO) country code elements as specified in ISO 3166-1-alpha-2.

Source

P. Emmenegger. Job Security Regulations in Western Democracies: A Fuzzy Set Analysis. *European Journal of Political Research*, 50(3):336-364, 2011.

eqmcc	<i>Minimize Canonical Sums using the Enhanced Quine-McCluskey Algorithm</i>
-------	---

Description

This function is the core function of the QCA package, which performs the reduction of canonical sums to minimal sums. It is called "eqmcc" because it is an 'e'nhancement of the classical Quine McCluskey minimization algorithm.

Usage

```
eqmcc(mydata, outcome = "", neg.out = FALSE, conditions = c(""), n.cut = 1,
      incl.cut1 = 1, incl.cut0 = 1, explain = "1", include = c(""), omit = c(),
      direxp = c(), rowdom = TRUE, details = FALSE, show.cases = FALSE,
      use.tilde = FALSE, use.letters = FALSE)
```

```
is.qca(x)
```

Arguments

mydata	A truth table object or a dataset of (binary or multi-value) crisp or fuzzy-set data
outcome	The name of the outcome set
neg.out	Logical, use negation of outcome set (ignored if mydata is a truth table object)
conditions	The names of the condition sets (if not specified, all sets in mydata but the outcome)

<code>n.cut</code>	The minimum number of cases with membership > 0.5 for an outcome value of "0", "1" or "C"
<code>incl.cut1</code>	The minimum sufficiency inclusion score for an outcome value of "1"
<code>incl.cut0</code>	The maximum sufficiency inclusion score for an outcome value of "0"
<code>explain</code>	The outcome value to be explained, either "1", "0" or "C"
<code>include</code>	The outcome value(s) of additional configurations to be included in the minimization
<code>omit</code>	A vector or a matrix of configurations to be omitted from minimization
<code>direxp</code>	A vector of directional expectations
<code>rowdom</code>	Logical, apply row dominance principle to eliminate dominated PIs
<code>details</code>	Logical, present details about solution
<code>show.cases</code>	Logical, also print case names if <code>details = TRUE</code>
<code>use.tilde</code>	Logical, use a tilde for set negation with binary-value set data
<code>use.letters</code>	Logical, use letters instead of set names
<code>x</code>	An object of class "qca"

Details

`mydata` can be a truth table object (an object of class "tt" returned by `truthTable()`) or a dataset of (binary or multi-value) crisp or fuzzy-set data. The dataset specified as `mydata` has to have the following structure: values of 0 and 1 for binary-value crisp sets, values between 0 and 1 for fuzzy set data, and values beginning with 0 and increments of 1 for multi-value crisp set data. "Don't care" values are indicated by a dash "-" or the placeholder "dc" or any negative integer. These values are ignored in the minimization. Sets which contain these values are excluded from the computation of parameters of fit.

If the argument `conditions` is not specified, all sets in the dataset of (binary or multi-value) crisp or fuzzy-set data but the outcome are included.

Configurations that contain fewer than `n.cut` cases with membership above 0.5 are coded as logical remainders ("?"). If the number of such cases is at least `n.cut`, configurations with an inclusion score of at least `incl.cut1` are coded as true ("1"), configuration with an inclusion score below `incl.cut1` but with at least `incl.cut0` are coded as a contradiction ("C"), and configurations with an inclusion score below `incl.cut0` are coded as false ("0"). If `incl.cut0` is not specified, it is set equal to `incl.cut1` and no contradictions can occur.

The argument `omit` can be used to omit any configuration from the minimization. If the `omit` argument is a vector, it should contain row numbers from the (complete) truth table. If it is a matrix, it should be of the same format as the truth table.

Directional expectations for filtering logical remainders are specified by the `direxp` argument. For binary-value crisp sets and fuzzy sets they should be given as a vector of the same length and in the same order of conditions as in `conditions`. A value of "1" indicates that the presence of the condition is expected to contribute to an outcome value of "1", "0" that the absence of the condition is expected to contribute to an outcome value of "1", and "-1" indicates no directional expectation regarding the set-theoretic relation between the configuration and the outcome set. In the case of multi-value crisp sets, directional expectations require the identification of the category name(s), separated by semicolons and all enclosed by double quotes. The provision of a category name

indicates that the presence of this category is expected to contribute to an outcome value of "1". Implicitly, it is the absence of all other categories which is expected to contribute.

If alternative minimal sums exist, all of them are printed if the row dominance principle for PIs is not applied as specified in the logical argument rowdom. One inessential prime implicant $P1$ dominates another $P2$ if all fundamental products covered by $P2$ are also covered by $P1$ and both are not interchangeable. Inessential PIs are listed in brackets in the solution output and at the end of the PI part in the parameters-of-fit table when `details = TRUE`, together with their unique coverage scores under each individual minimal sum.

If the conditions are already named with single letters, the argument `use.letters` will have no effect.

Value

An invisible list with the following components:

<code>tt</code>	the truth table object
<code>excluded</code>	the line numbers of the excluded configurations
<code>initials</code>	the initial fundamental products to be explained
<code>PIs</code>	the Prime Implicants, results of the minimization procedure
<code>PIchart</code>	a list containing the PI chart(s)
<code>solution</code>	a list of solution(s)
<code>SA</code>	the Simplifying Assumptions

References

- A. Dusa. A Mathematical Approach to the Boolean Minimization Problem. *Quality & Quantity*, 44(1), pp.99-113, 2010.
- A. Dusa. *Enhancing Quine-McCluskey*. WP 2007-49, COMPASSS, 2007.
URL: <http://www.compasss.org/files/WPfiles/Dusa2007a.pdf>.
- P. Emmenegger. Job Security Regulations in Western Democracies: A Fuzzy Set Analysis. *European Journal of Political Research*, 50(3):336-364, 2011.
- C. Hartmann and J. Kemmerzell. Understanding Variations in Party Bans in Africa. *Democratization*, 17(4):642-665, 2010.
- M.L. Krook. Women's Representation in Parliament: A Qualitative Comparative Analysis. *Political Studies*, 58(5):886-908, 2010.
- C.C. Ragin. *Redesigning Social Inquiry: Fuzzy Sets and Beyond*. University of Chicago Press, Chicago, 2008.
- C.C. Ragin and S.I. Strand. Using Qualitative Comparative Analysis to Study Causal Order: Comment on Caren and Panofsky (2005). *Sociological Methods & Research*, 36(4):431-441, 2008.

See Also

[truthTable](#)

Examples

```

# csQCA using Krook (2010)
#-----
data(Krook)
Krook

# explain true configurations, complex solution
eqmcc(Krook, outcome = "WNP")

# explain true configurations with negated outcome set, complex solution
eqmcc(Krook, outcome = "WNP", neg.out = TRUE)

# same result with false configurations (not always the case!)
eqmcc(Krook, outcome = "WNP", explain = "0")

# explain true configurations, parsimonious solution,
# with solution details
eqmcc(Krook, outcome = "WNP", include = "?", details = TRUE)

# explain true configurations, parsimonious solution,
# with solution details and without row dominance
KrookSP <- eqmcc(Krook, outcome = "WNP", include = "?", details = TRUE,
  rowdom = FALSE)
KrookSP

# pass truth table object to eqmcc() and derive complex solution
KrookTT <- truthTable(Krook, outcome = "WNP")
KrookSC <- eqmcc(KrookTT)
KrookSC

# print fundamental products
KrookSC$initials

# fsQCA using Emmenegger (2011)
#-----
data(Emme)
Emme

# explain false configurations, parsimonious solution,
# with solution details
eqmcc(Emme, outcome = "JSR", explain = "0", include = "?",
  details = TRUE)

# explain true configurations, intermediate solution,
# with directional expectations and solution details
EmmeSI <- eqmcc(Emme, outcome = "JSR", incl.cut1 = 0.9, include = "?",
  direxp = c(1,1,1,1,1,0), details = TRUE)
EmmeSI

# check PI chart for intermediate solution
EmmeSI$PIchart$i.sol

```

```

# mvQCA using Hartmann and Kemmerzell (2010)
#-----
data(HarKem)
HarKem

conds <- c("C", "F", "T", "V")

# explain true configurations, parsimonious solution,
# with contradictions
HarKemSP <- eqmcc(HarKem, outcome = "PB", conditions = conds,
  include = c("?", "C"))
HarKemSP

# explain the contradictions
# N.B.: Only one contradiction, no minimization
eqmcc(HarKem, outcome = "PB", conditions = conds, incl.cut0 = 0.4,
  explain = "C")

# explain true configurations, intermediate solution,
# with directional expectations:
# C{1}, F{1,2}, T{2}, V contribute to PB

HarKemSI <- eqmcc(HarKem, outcome = "PB", conditions = conds,
  include = "?", direxp = c(1, "1;2", 2, 1))
HarKemSI

# tQCA using Ragin and Strand (2008)
#-----
data(RagStr)
RagStr

# explain true configurations, complex solution,
# with solution details and cases;
# condition EBA is automatically excluded from parameters of fit
eqmcc(RagStr, outcome = "REC", details = TRUE, show.cases = TRUE)

```

factorize

Factorize a QCA Solution

Description

This function finds all combinations of common factors of literals in a solution.

Usage

```
factorize(sol.obj, splitmethod = "", sort.by.literals = FALSE, sort.by.number = FALSE)
```

Arguments

sol.obj An object containing either the list of solutions, or a single character solution

splitmethod The separator of the literals within a Boolean product

sort.by.literals Logical, sort results by the largest number of literals as common factor

sort.by.number Logical, sort results by the largest number of elements that have been factorized

References

C. C. Ragin. *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987.

See Also

[eqmcc](#)

Examples

```
factorize("AB + AC + CD")

factorize("one*TWO*four + one*THREE + THREE*four", splitmethod = "*")

# factorize solution object directly
data(Emme)
sol <- eqmcc(Emme, outcome = "JSR", incl.cut1 = 0.9)
factorize(sol)

# sort by largest number of factoring literals
factorize(sol, sort.by.literals = TRUE)

# sort by largest number of factorized elements
factorize(sol, sort.by.number = TRUE)
```

findSubsets

Find all Possible Subsets of a Given Prime Implicant

Description

It is a general rule that all subsets can be found in the n^k space, understood as all possible combinations of values in any combination of bases n , each causal condition having three or more levels (Dusa, 2007, 2010). If a prime implicant can be considered a superset of an initial combination of causal conditions, the reverse is also true: the initial combination is a subset of a prime implicant. Even more, a less minimum prime implicant (with more literals) is also a subset of another shorter (more minimum) prime implicant. This function finds all possible such subsets for a given prime implicant, in the n^k space.

Usage

```
findSubsets(noflevels3k, row.no, maximum)
```

Arguments

noflevels3k	A vector containing the number of levels for each causal condition plus 1 (because all subsets are to be found in the higher "3k" matrix)
row.no	The row number where the (minimum) prime implicant is located
maximum	The maximum line number (subset) to be returned

Value

A vector with the line numbers of all possible subsets in the $prod(n)$ space.

References

A. Dusa. *Enhancing Quine-McCluskey*. WP 2007-49, COMPASSS, 2007.
 URL: <http://www.compass.org/files/WPfiles/Dusa2007a.pdf>.

A. Dusa. A Mathematical Approach to the Boolean Minimization Problem. *Quality & Quantity*, 44(1): 99-113, 2010.

See Also

[findSupersets](#)

Examples

```
# all three conditions are binary, having two levels: 0 and 1
noflevels <- c(2, 2, 2)

# 5 8 11 14 17 20 23 26
findSubsets(noflevels + 1, 2)

# stopping at maximum row number 20
# 5 8 11 14 17 20
findSubsets(noflevels + 1, 2, 20)
```

findSupersets

Find all Possible Supersets of One or Several Combinations

Description

It is a general rule that all supersets can be found in the n^k space, understood as all possible combinations of values in any combination of bases n , each causal condition having three or more levels (Dusa, 2010, 2007). There is a finite number of prime implicants for any combination of causal conditions, equal to $2^k - 1$, where k is the number of causal conditions. Counting out the input combination itself, the actual number is $2^k - 2$. This function computes all possible unique supersets for a specific set of combinations (either to explain or to exclude).

Usage

```
findSupersets(noflevels3k, input.combs)
```

Arguments

noflevels3k	A vector containing the number of levels for each causal condition plus 1 (because all prime implicants are to be found in the higher matrix)
input.combs	A matrix with combinations of causal conditions or a vector of line numbers from the same matrix

Value

a vector with the line numbers of all possible prime implicants in the 3^k space

References

A. Dusa. *Enhancing Quine-McCluskey*. WP 2007-49, COMPASSS, 2007.

URL: <http://www.compass.org/files/WPfiles/Dusa2007a.pdf>.

A. Dusa. A Mathematical Approach to the Boolean Minimization Problem. *Quality & Quantity*, 44(1): 99-113, 2010.

See Also

[getRow](#), [findSubsets](#)

Examples

```
# all three conditions are binary, having two levels: 0 and 1
noflevels <- c(2, 2, 2)

# 2 4 5 10 11 13 14
findSupersets(noflevels + 1, 14)

# 2 7 8 10 11 16 17
findSupersets(noflevels + 1, 17)

# both line numbers 14 and 17
# 2 4 5 7 8 10 11 13 14 16 17
findSupersets(noflevels + 1, c(14, 17))

# input.combs as a matrix
(input.combs <- getRow(noflevels + 1, c(14, 17)))

#      [,1] [,2] [,3]
#[1,]   1   1   1
#[2,]   1   2   1

# 2 4 5 7 8 10 11 13 14 16 17
findSupersets(noflevels + 1, input.combs)
```

`findTh`*Find Calibration Thresholds*

Description

This function finds calibration thresholds for splitting base variables into the desired number of groups using cluster analysis.

Usage

```
findTh(x, groups = 2, hclustm = "complete", distm = "euclidean")
```

Arguments

<code>x</code>	An interval or ratio-scaled base variable
<code>groups</code>	A vector of integers with the desired number of groups, see <code>?cutree</code>
<code>hclustm</code>	The agglomeration (clustering) method to be used, see <code>?hclust</code>
<code>distm</code>	The distance measure to be used, see <code>?dist</code>

Value

A numeric vector of suggested threshold(s) for dividing base variables into the desired number of groups.

Note

Default values from the `hclust()` method and the `dist()` method are used for both the distance measure `distm` and the clustering method `hclustm`. Some prefer to use "average" for the `hclust()` method, but results are generally the same.

See Also

[cutree](#), [hclust](#), [dist](#)

Examples

```
# 15 random values between 1 and 100
x <- sample(1:100, size = 15)

# split into 2 groups for csQCA
findTh(x)

# split into 3 groups for mvQCA
findTh(x, groups = 3)
```

getRow	<i>Get a Specific Row from a Truth Table</i>
--------	--

Description

A truth table is formed by the unique combinations of causal conditions' levels. For binary crisp sets, there are exactly 2^k combinations, where k is the number of conditions. This function accepts multiple levels (not just 0 and 1) and its purpose is to transform the decimal representation of a row number into its corresponding combination of levels.

Usage

```
getRow(noflevels, row.no, zerobased = FALSE)
```

Arguments

noflevels	A vector containing the number of levels for each condition
row.no	A vector, the desired row number(s) from the truth table
zerobased	Logical, the first row number from the truth table is zero

Value

a matrix with the combination of levels corresponding to the desired row number(s)

See Also

[createMatrix](#), [expand.grid](#)

Examples

```
# all three conditions are binary, having two levels: 0 and 1
noflevels <- c(2, 2, 2)
getRow(noflevels, 8) # 1 1 1
getRow(noflevels, 0, zerobased=TRUE) # 0 0 0

# the second condition has three levels: 0, 1 and 2
noflevels <- c(2, 3, 2)
getRow(noflevels, 11) # 1 2 0
```

HarKem	<i>Introduction of Party Ban Provisions and their Implementation in sub-Saharan Africa</i>
--------	--

Description

The data frame **HarKem** has 48 rows and 7 columns.

This data is from the study by Hartmann and Kemmerzell (2010), which analyzes the determinants of the introduction of party ban provisions and their actual implementation in sub-Saharan Africa using mvQCA.

Usage

```
data(HarKem)
```

Format

The data contains the following sets:

- C** Condition: colonial tradition ("2" British, "1" French, "0" other)
- F** Condition: former regime type competition ("2" no, "1" limited, "0" multi-party)
- T** Condition: mode of transition ("2" managed, "1" pacted, "0" democracy before 1990)
- R** Condition: regime type ("2" stable authoritarian, "1" liberalizing/blocked, "0" democratic/democratizing)
- V** Condition: ethnic violence ("1" yes, "0" no)
- PB** Outcome : party ban provisions introduced ("1" yes, "0" no)
- PBI** Outcome : party bans implemented ("1" yes, "0" no)

Note

The row names are the official International Organization for Standardization (ISO) country code elements as specified in ISO 3166-1-alpha-2.

Source

C. Hartmann and J. Kemmerzell. Understanding Variations in Party Bans in Africa. *Democratization*, 17(4):642-665, 2010.

Krook	<i>Women's Representation in Western-Democratic Parliaments</i>
-------	---

Description

The data frame **Krook** has 22 rows and 6 columns.

This data is from the study by Krook (2010), which analyzes the determinants of high women's representation in Western-democratic parliaments using csQCA.

Usage

data(Krook)

Format

The data contains the following sets:

- ES** Condition: PR electoral system ("1" yes, "0" no)
- QU** Condition: quota for women ("1" yes, "0" no)
- WS** Condition: social-democratic welfare system ("1" yes, "0" no)
- WM** Condition: autonomous women's movement ("1" yes, "0" no)
- LP** Condition: seats held by left-libertarian parties ("1" at least 7 percent, "0" less than 7 percent)
- WNP** Outcome : women in the single or lower house of parliament ("1" at least 30 percent, "0" less than 30 percent)

Note

The row names are the official International Organization for Standardization (ISO) country code elements as specified in ISO 3166-1-alpha-2.

Source

M. L. Krook. Women's Representation in Parliament: A Qualitative Comparative Analysis. *Political Studies*, 58(5):886-908, 2010.

pof

Set-Relational Parameters of Fit

Description

This function computes three set-relational parameters of fit: inclusion, PRI and coverage scores.

Usage

```
pof(setms, mydata, outcome = "", neg.out = FALSE, relation = "necessity", ...)
```

Arguments

- | | |
|----------|--|
| setms | Set membership scores, or a matrix with crisp set representations, or a vector of line numbers |
| mydata | The original dataset |
| outcome | The name of the outcome set |
| neg.out | Logical, use negation of outcome set |
| relation | The set relation of the conditions to the outcome, either "necessity" or "sufficiency" |
| ... | Other arguments from the generic print (not used in this function) |

Details

The argument `setms` specifies a dataframe of *set* membership scores, where *set* refers to any kind of set, including original condition sets, conjunctive or disjunctive combinations returned by `superSubset()`, prime implicants returned by `eqmcc()` or any other compound set.

It can also be a matrix with the crisp set representation of the causal conditions, or even a vector of line numbers corresponding to those representations.

References

P. Emmenegger. Job Security Regulations in Western Democracies: A Fuzzy Set Analysis. *European Journal of Political Research*, 50(3):336-364, 2011.

M. L. Krook. Women's Representation in Parliament: A Qualitative Comparative Analysis. *Political Studies*, 58(5):886-908, 2010.

Examples

```
# csQCA using Krook (2010)
#-----
data(Krook)
Krook
x <- Krook[,1:5]

# get necessity parameters of fit for all conditions
pof(x, Krook, outcome = "WNP")

# now for the negated outcome
pof(x, Krook, outcome = "WNP", neg.out = TRUE)

# get sufficiency parameters of fit for all original conditions
pof(x, Krook, outcome = "WNP", relation = "sufficiency")

# now for the negated outcome
pof(x, Krook, outcome = "WNP", neg.out = TRUE, relation = "sufficiency")

# fsQCA using Emmenegger (2011)
#-----
data(Emme)
Emme

# first test for necessary conditions with superSubset(), then
# check whether the returned combinations are also necessary for
# the negation of the outcome

EmmeNR <- superSubset(Emme, outcome = "JSR", incl.cut = 0.965, cov.cut = 0.6)
EmmeNR
pof(EmmeNR$coms, Emme, outcome = "JSR", neg.out = TRUE)

# first derive the complex solution, then check whether the negations
# of the prime implicants are also sufficient for the outcome

EmmeSC <- eqmcc(Emme, outcome = "JSR", incl.cut1 = 0.9, details = TRUE)
```

```

EmmeSC
pof(1 - EmmeSC$pims$c.sol, Emme, outcome = "JSR", relation = "sufficiency")

# parameters of fit for any term, including configurations;
# use "-1" as a placeholder for a minimized literal;
#      [,1] [,2] [,3] [,4] [,5] [,6]
#[1,]  -1  -1  -1   1   0   1
#[2,]   1   0   1  -1   1   0

confs <- matrix(c(-1, -1, -1, 1, 0, 1,
                  1, 0, 1, -1, 1, 0), nrow = 2, byrow = TRUE)
confs
pof(confs, Emme, outcome = "JSR", relation = "sufficiency")

# or even vectors of line numbers from the n^k matrix
pof(c(43, 57), Emme, "JSR", relation = "sufficiency")

# in this case, the line numbers 43 and 57 represent the following
# terms / configurations:
#      [,1] [,2] [,3] [,4] [,5] [,6]
#[1,]  -1  -1   0   0   1  -1  -> l*r*P
#[2,]  -1  -1   1  -1  -1   1  -> L*V

getRow(rep(3, 6), c(43, 57)) - 1

```

Description

The data frame **RagStr** has 17 rows and 6 columns.

This data is originally from the hypothetical study by Caren and Panofsky (2005), which analyzes the determinants of unionization attempts by graduate student workers at research universities using tQCA. The data has been reanalyzed in the study by Ragin and Strand (2008).

Usage

```
data(RagStr)
```

Format

The data contains the following sets:

- P** Condition: public university ("1" yes, "0" no)
- E** Condition: support of elite allies ("1" yes, "0" no)
- A** Condition: national union affiliation ("1" yes, "0" no)
- S** Condition: a strike or strike threat ("1" yes, "0" no)
- EBA** Condition: **E** present before **A** ("1" yes, "0" no, "dc" don't care)
- REC** Outcome : union recognition ("1" yes, "0" no)

Source

<http://www.compass.org>

References

N. Caren and A. Panofsky. TQCA: A Technique for Adding Temporality to Qualitative Comparative Analysis. *Sociological Methods & Research*, 34(2):147-172, 2005.

C. C. Ragin and S. I. Strand. Using Qualitative Comparative Analysis to Study Causal Order: Comment on Caren and Panofsky (2005). *Sociological Methods & Research*, 36(4):431-441, 2008.

readTosmana

Read Tosmana XML Data File

Description

Tosmana can save data in XML format. This function reads XML files and coerces them to R data frame objects.

Usage

```
readTosmana(filename)
```

Arguments

filename A string, the path to the XML file

Note

This function requires the XML package

Rokkan

Divided Working-Class Movements

Description

The data frame **Rokkan** has 16 rows and 5 columns.

Abridged from Ragin (1987: 129):

The data was used by Rokkan (1970) in his work on nation building in Western Europe. Rokkan used a "configurational" approach that bears many similarities to the Boolean approach presented in this work. His main substantive interest was the growth of mass democracy and the emergence of different cleavage structures in Western European polities. One outcome that interested him was the division of some working-class movements in these countries following the Russian Revolution into internationally oriented wings and some into nationally oriented wings. He considered the distribution of this outcome important because of its implication for the future of working-class mobilization (and cleavage structures in general) in Western Europe.

Usage

```
data(Rokkan)
```

Format

The dataset contains the following sets:

- C** Condition: National church ("1" yes, "0" no)
- R** Condition: Significant Roman Catholic population and participation in mass education ("1" yes, "0" no)
- L** Condition: State protection of landed interests ("1" yes, "0" no)
- E** Condition: Early state ("1" yes, "0" no)
- S** Outcome : Major split in working-class movement provoked by Russian Revolution ("1" yes, "0" no)

Note

The row names are the official International Organization for Standardization (ISO) country code elements as specified in ISO 3166-1-alpha-2.

Source

C. C. Ragin. *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987.

References

S. Rokkan. *Citizens, Elections, Parties*. McKay, New York, 1970.

solveChart

Reduce the Redundant Prime Implicants

Description

While the minimization procedure finds a certain number of prime implicants, not all of them are necessary. This function reduces the number of prime implicants to a minimal solution.

Usage

```
solveChart(chart)
```

Arguments

chart The prime implicants chart, a matrix with TRUE/FALSE values

Value

a matrix containing row indices of all possible solutions

References

C. C. Ragin. *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987.

See Also

[createChart](#)

Examples

```
# the chart to be reduced
chart <- demoChart(c("A", "B", "c"), c("ABC", "Abc", "AbC", "aBc"))
prettyTable(chart)

# solution: first and second or first and third prime implicants
solveChart(chart)
```

superSubset

Find Superset/Subset Relations

Description

This function finds all conjunctive/disjunctive combinations of conditions among all possible such combinations which optimize the fulfilment of the specified criteria set for a superset (necessity) / subset (sufficiency) relation to the outcome and are minimally complex.

Usage

```
superSubset(mydata, outcome = "", neg.out = FALSE, conditions = c(""),
            relation = "necessity", incl.cut = 1, cov.cut = 0, use.tilde = FALSE,
            use.letters = FALSE)
```

Arguments

mydata	A dataset of (binary or multi-value) crisp or fuzzy-set data
outcome	The name of the outcome set
neg.out	Logical, use negation of outcome set
conditions	The names of the condition
relation	The set relation of the conditions to the outcome, either "necessity" or "sufficiency"
incl.cut	The minimal inclusion score
cov.cut	The minimal coverage score
use.tilde	Logical, use a tilde for set negation
use.letters	Logical, use letters instead of set names

Details

This function returns a list of those combinations of the $3^k - 1$ potential combinations of k conditions which are minimally complex, yet exceed the given cut-offs for inclusion and coverage scores. In addition to inclusion and coverage scores, PRI (proportional reduction in inconsistency) scores are also returned. The following paragraphs describe the theoretical idea behind the function. Computationally, the algorithm implementation is more efficient.

If `relation = "necessity"`, the initial set of combinations is comprised of the 2^k order-1 combinations as well as their negations. This set is expanded incrementally by forming conjunctive combinations of a higher order as long as the cut-offs are still met. The inclusion cut-off always enjoys a higher priority than the coverage cut-off. All resulting combinations as well as all subcombinations of a lower order will be returned.

If none of the initial order-1 combinations passes the inclusion cut-off, `superSubset()` will search for disjunctive combinations until the cut-offs have been met. Only the disjunctive combinations thus found will be returned.

If `relation = "sufficiency"`, the initial set of combinations is comprised of all order- k conjunctive combinations. This set is reduced incrementally by forming conjunctive combinations of a lower order as long as the cut-offs are still met. Only the conjunctive combinations of the lowest order will be printed. Disjunctive combinations are not formed. For more details, see Thiem and Dusa (2012).

The argument `use.tilde` only applies to non-multi-value data. If the conditions are already named with single letters, the argument `use.letters` will have no effect.

Value

An object of class "ss", which is a list with the following components:

```
$incl.cov:  a dataframe with the parameters of fit
$coms:     a dataframe with the combination membership scores
```

References

- P. Emmenegger. Job Security Regulations in Western Democracies: A Fuzzy Set Analysis. *European Journal of Political Research*, 50(3):336-364, 2011.
- M. L. Krook. Women's Representation in Parliament: A Qualitative Comparative Analysis. *Political Studies*, 58(5):886-908, 2010.
- A. Thiem and A. Dusa. *Qualitative Comparative Analysis with R: A User's Guide*. Springer, New York, 2012.

Examples

```
# csQCA using Krook (2010)
#-----
data(Krook)
Krook

# find all optimal combinations with a necessity inclusion score
```

```

# of at least 0.9 and a necessity coverage score of at least 0.6
KrookSS <- superSubset(Krook, outcome = "WNP", incl.cut = 0.9, cov.cut = 0.6)
KrookSS

# the combination membership scores for all cases
KrookSS$coms

# fsQCA using Emmenegger (2011)
#-----
data(Emme)
Emme

# find all optimal combinations with a sufficiency inclusion score
# of at least 0.9 and sufficiency coverage score of at least 0.4
# N.B.: Some combinations may appear as solution PIs.
EmmeSS <- superSubset(Emme, outcome = "JSR", relation = "sufficiency",
  incl.cut = 0.9, cov.cut = 0.4)
EmmeSS

# the same criteria, but for the negation of the outcome
EmmeSSN <- superSubset(Emme, outcome = "JSR", neg.out = TRUE,
  relation = "sufficiency", incl.cut = 0.9, cov.cut = 0.4)
EmmeSSN

# the combination membership scores for all cases
EmmeSSN$coms

```

truthTable

Create a Truth Table

Description

A truth table lists every *configuration* (combination of condition literals) and its outcome value. This function creates truth tables from any kind of common input data. For crisp-set data (binary and multi-valued), cases are assigned directly to the unique configuration in which they have full membership. For fuzzy-set data, the function first detects in which configuration a case has membership above 0.5 before assigning it to this configuration.

`is.tt()` checks whether an object has class "tt" (truth table); such an object is created by `truthTable()`

Usage

```

truthTable(mydata, outcome = "", neg.out = FALSE, conditions = c(""), n.cut = 1,
  incl.cut1 = 1, incl.cut0 = 1, complete = FALSE, show.cases = FALSE,
  sort.by = c(""), decreasing = TRUE, use.letters = FALSE, ...)

```

```
is.tt(x)
```

Arguments

<code>mydata</code>	A dataset of (binary or multi-valued) crisp or fuzzy-set data
<code>outcome</code>	The name of the outcome set
<code>neg.out</code>	Logical, use negation of outcome set
<code>conditions</code>	The names of the conditions
<code>n.cut</code>	The minimum number of cases with membership > 0.5 for an outcome value of "0", "1" or "C"
<code>incl.cut1</code>	The minimum sufficiency inclusion score for an outcome value of "1"
<code>incl.cut0</code>	The maximum sufficiency inclusion score for an outcome value of "0"
<code>complete</code>	Logical, print complete truth table
<code>show.cases</code>	Logical, print case names
<code>sort.by</code>	Sort truth table by inclusion scores and/or number of cases
<code>decreasing</code>	Sort in decreasing or increasing order of value(s) passed to <code>sort.by</code>
<code>use.letters</code>	Logical, use letters instead of set names
<code>x</code>	An object of class "tt"
<code>...</code>	Other arguments (used internally)

Details

The dataset specified as `mydata` has to have the following structure: values of 0 and 1 for binary-value crisp sets, values between 0 and 1 for fuzzy set data, and values beginning with 0 and increments of 1 for mutli-value crisp set data. "Don't care" values are indicated by a dash "-", the placeholder "dc" or any negative integer. These values are ignored in the minimization. Sets which contain these values are excluded from the computation of parameters of fit.

If the argument `conditions` is not specified, all sets in `mydata` but the outcome are included as `conditions`.

Configurations that contain fewer than `n.cut` cases with membership above 0.5 are coded as logical remainders ("?"). If the number of such cases is at least `n.cut`, configurations with an inclusion score of at least `incl.cut1` are coded as true ("1"), configuration with an inclusion score below `incl.cut1` but with at least `incl.cut0` are coded as a contradiction ("C"), and configurations with an inclusion score below `incl.cut0` are coded as false ("0"). If `incl.cut0` is not specified, it is set equal to `incl.cut1` and no contradictions can occur.

The `sort.by` argument orders all configurations by their inclusion scores (*incl*) or the number of cases with membership above 0.5 they contain (*n*) or both, in either order.

If the `conditions` are already named with single letters, the argument `use.letters` will have no effect.

Value

An object of class "tt", which is a list with the following components:

<code>tt</code>	the truth table itself
-----------------	------------------------

indexes	the truth table line numbers
noflevels	a vector with the number of levels for all conditions
initial.data	the initial data
recoded.data	the recoded data, for post fuzzy-set conversion
cases	case names for the truth table combination

References

- P. Emmenegger. Job Security Regulations in Western Democracies: A Fuzzy Set Analysis. *European Journal of Political Research*, 50(3):336-364, 2011.
- C. Hartmann and J. Kemmerzell. Understanding Variations in Party Bans in Africa. *Democratization*, 17(4):642-665, 2010.
- M. L. Krook. Women's Representation in Parliament: A Qualitative Comparative Analysis. *Political Studies*, 58(5):886-908, 2010.
- C. C. Ragin. *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987.
- C. C. Ragin. *Redesigning Social Inquiry: Fuzzy Sets and Beyond*. University of Chicago Press, Chicago, 2008.
- C. C. Ragin and S. I. Strand. Using Qualitative Comparative Analysis to Study Causal Order: Comment on Caren and Panofsky (2005). *Sociological Methods & Research*, 36(4):431-441, 2008.
- A. Thiem and A. Dusa. *Qualitative Comparative Analysis with R: A User's Guide*. Springer, New York, 2012.

See Also

[eqmcc](#)

Examples

```
# csQCA using Krook (2010)
#-----
data(Krook)
Krook

# print the truth table
truthTable(Krook, outcome = "WNP")

# print the complete truth table, show cases, sort by inclusions
# scores and then number of cases
truthTable(Krook, outcome = "WNP", complete = TRUE, show.cases = TRUE,
  sort.by = c("incl", "n"))

# code configurations with 1 case as remainders
KrookTT <- truthTable(Krook, outcome = "WNP", n.cut = 2, show.cases = TRUE)
KrookTT

# print the cases that were assigned to remainders based on n.cut
```

```
KrookTT$excluded

# fsQCA using Emmenegger (2011)
#-----
data(Emme)
Emme

# code non-remainder configurations with inclusion scores between 0.5
# and 0.9 as contradictions
EmmeTT <- truthTable(Emme, outcome = "JSR", incl.cut1 = 0.9, incl.cut0 = 0.5)
EmmeTT

# truth table based on negated outcome set
EmmeTT <- truthTable(Emme, outcome = "JSR", neg.out = TRUE, incl.cut1 = 0.9,
  incl.cut0 = 0.5)
EmmeTT

# get the raw truth table as a component of the truth table object EmmeTT
EmmeTT$tt

# mvQCA using Hartmann and Kemmerzell (2010)
#-----
data(HarKem)
HarKem

# code non-remainder configurations with inclusion scores below 1 but above
# 0.4 as contradictions
HarKemTT <- truthTable(HarKem, outcome = "PB",
  conditions = c("C","F","T","V"), incl.cut0 = 0.4)
HarKemTT

# list the number of condition levels
HarKemTT$noflevels

# tQCA using Ragin and Strand (2008)
#-----
data(RagStr)
RagStr

# tQCA truth table with "don't care" values
truthTable(RagStr, outcome = "REC")
```

Index

*Topic **datasets**

Emme, 9
HarKem, 20
Krook, 20
RagStr, 23
Rokkan, 24

*Topic **functions**

allExpressions, 3
calibrate, 4
createMatrix, 7
demoChart, 8
eqmcc, 10
factorize, 14
findSubsets, 15
findSupersets, 16
findTh, 18
getRow, 19
pof, 21
readTosmana, 24
solveChart, 25
superSubset, 26
truthTable, 28

*Topic **package**

QCA-package, 2

allExpressions, 3, 7

calibrate, 4
createChart, 26
createMatrix, 4, 7, 19
cutree, 18

demoChart, 8
dist, 18

Emme, 9
eqmcc, 10, 15, 30
expand.grid, 19

factorize, 14
findSubsets, 15, 17

findSupersets, 16, 16

findTh, 18

getRow, 17, 19

HarKem, 20
hclust, 18

is.qca (eqmcc), 10
is.tt (truthTable), 28

Krook, 20

pof, 21
prettyTable, 9

QCA (QCA-package), 2
QCA-package, 2

RagStr, 23
readTosmana, 24
Rokkan, 24

solveChart, 25
superSubset, 26

truthTable, 7, 12, 28