QCA: A Package for Qualitative Comparative Analysis

by Alrik Thiem and Adrian Duşa

Abstract We present QCA, a package for performing Qualitative Comparative Analysis (QCA). QCA is becoming increasingly popular with social scientists, but none of the existing software alternatives covers the full range of core procedures. This gap is now filled by QCA. After a mapping of the method’s diffusion, we introduce some of the package’s main capabilities, including the calibration of crisp and fuzzy sets, the analysis of necessity relations, the construction of truth tables and the derivation of complex, parsimonious and intermediate solutions.

Introduction

Qualitative Comparative Analysis (QCA) - a research method popularized largely through the work of Charles Ragin (Ragin, 1987, 2000, 2008) - counts among the most influential recent innovations in social science methodology. In line with Ragin’s own background, QCA has been initially employed only by a small number of (political) sociologists (e.g., Amenta et al., 1992; Griffin et al., 1991; Wickham-Crowley, 1991). Since then, however, the method has made inroads into political science and international relations (e.g., Thiem, 2011; Vis, 2009), business and economics (e.g., Evans and Aligica, 2008; Valliere et al., 2008), management and organization (e.g., Greckhamer, 2011), legal studies and criminology (Arvind and Stirton, 2010; Miethe and Drass, 1999), education (e.g., Glaesser and Cooper, 2011; Schneider and Sadowski, 2010), and health policy research (e.g., Harkreader and Imershein, 1999; Schensul et al., 2010). Figure 1 charts the trend in the total number of QCA applications that have appeared in peer-reviewed journal articles since 1984, broken down by its three variants crisp-set QCA (csQCA), multi-value QCA (mvQCA) and fuzzy-set QCA (fsQCA).

Allowing for a publication lag of about two years, 4.2 applications on average have been published throughout the first decade following the introduction of csQCA in Ragin (1987). But only his sequel “Fuzzy-Set Social Science” (Ragin, 2000) seems to have got the “Ragin Revolution” (Vaisey, 2009) eventually off ground. In the years from 2003 to 2007, the average number of applications had risen to 13.6 before the absolute number of applications more than tripled from 12 in 2007 to 39 in 2011. Despite the introduction of fsQCA, applications of csQCA have continued to increase from four in 2001 to 22 in 2011. In contrast to csQCA and fsQCA, mvQCA has remained underutilized. Of a total of 280 applications between 1984 and 2012, only ten have employed this variant. Even when accounting for the fact that it has been introduced in 2004, 17 years after csQCA and four years after fsQCA, this represents a disproportionately low number.

QCA’s methodological success story has created a growing demand for tailored software, which

---

1 The number of applications differs slightly from the number of articles as four articles have each presented two applications of QCA using two different variants. In order to be included in the data, applications had to focus primarily on a substantive research question, not QCA as a method. All entries have been recorded in the bibliography section on http://www.compasss.org.

2 See the debate in Field Methods for more details on the status of mvQCA (Thiem, 2013; Vink and van Vliet, 2009, 2013).
has been met almost exclusively by two programmes: Charles Ragin and Sean Davey’s (2009) fs/QCA and Lasse Cronqvist’s (2011) Tosmana. Until recently, however, users of non-Windows operating systems were limited as neither programme ran on other operating systems than Microsoft Windows. As of version 1.3.2.0, Tosmana has also supported other operating systems. In 2008 and 2012, Kyle Longest and Stephen Vaisey’s (2008) fuzzy package for Stata and Christopher Reichert and Claude Rubinson’s (2013) Kirq have been developed as alternatives to fs/QCA. For the R environment, Adrian Duša’s QCA package has been first added in 2006 and in 2009, Ronggui Huang (2011) has released the QCA3 package. The detailed market shares of these software solutions are also shown in Figure 1.\textsuperscript{3} Holding a clear monopoly, fs/QCA is by far the most common software with 82%, followed by Tosmana with 14% and fuzzy with 1%. Other solutions have claimed about 3%, but neither R package has managed to win any market shares thus far.

Not as unequal as their market shares, but significantly different still, are the capabilities of these software solutions. Table 1 provides an overview of the functionality each programme offers. All alternatives to QCA have different capabilities, but none covers the entire range of basic procedures. Kirq, fs/QCA and fuzzy cannot handle multi-value sets, whereas Tosmana cannot process fuzzy sets. The possibility to analyze necessity relations is not implemented in Tosmana, either, and the other packages, except Kirq, offer only limited user-driven routines. Complex and parsimonious solutions can be found by all packages, but only fs/QCA generates intermediate solutions on an automated basis.

<table>
<thead>
<tr>
<th>Function</th>
<th>Tosmana\textsuperscript{b}</th>
<th>Kirq\textsuperscript{c}</th>
<th>fs/QCA\textsuperscript{d}</th>
<th>fuzzy\textsuperscript{e}</th>
<th>QCA3\textsuperscript{f}</th>
<th>QCA\textsuperscript{g}</th>
</tr>
</thead>
<tbody>
<tr>
<td>csQCA</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>mvQCA</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>fsQCA</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>(tQCA)</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>complex</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>intermediate</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>parsimonious</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>necessity tests</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>parameters of fit</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>calibration</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>factorization</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>identify (C)SAs</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>statistical tests</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
</tbody>
</table>

\textsuperscript{a} / / = no/partial/full functionality
\textsuperscript{b} version 1.3.2.0;\textsuperscript{c} version 2.1.9;\textsuperscript{d} version 2.5;\textsuperscript{e} version st0140_2;\textsuperscript{f} version 0.0-5;\textsuperscript{g} version 1.0-5

The calibration of crisp sets is limited in fs/QCA and QCA3. Tosmana cannot handle fuzzy sets, but it provides more elaborate tools for the calibration of crisp sets. In addition to Ragin’s (2008) “direct method” and “indirect method”, fuzzy offers a set-normalizing linear membership function. Most importantly, it also includes various statistical procedures for coding truth tables, the appropriateness of which largely depends on the research design.\textsuperscript{4}

QCA combines and enhances the individual strengths of other software solutions. It can process all QCA variants (including temporal QCA (tQCA)), generates all solution types, and offers a wide range of procedures. For example, QCA provides four classes of functions for almost all kinds of calibration requirements and has an automated routine for the analysis of necessity relations. QCA is also the only package that can factorize any Boolean expression. As in Tosmana and QCA3, simplifying assumptions can also be identified. Unlike Tosmana, however, which does not present any parameters of fit, QCA produces inclusion, coverage and PRI (Proportional Reduction in Inconsistency) scores for

\textsuperscript{3}Only 156 out of 276 articles cite the software program that was used.
\textsuperscript{4}As QCA has been initially developed for medium-\(n\) (\(\approx\)10-30 cases) macro-level data analysis, often with universes of objects rather than random samples, statistical tests are not only often meaningless but also rarely produce statistically significant configurations. With large-\(n\) micro-level data, such tests may be more appropriate.
both necessity and sufficiency relations in mvQCA.\textsuperscript{5}

In summary, a comprehensive QCA software solution has been missing so far. Researchers have often been limited in their analyses when using one programme, or they had to resort to different programmes for performing all required operations. This gap is now filled by the QCA package, which seeks to provide a user-friendly yet powerful command-line interface alternative to the two dominant graphical user interface solutions fs/QCA and Tosmana. In the remainder of this article, we introduce some of the package’s most important functions, including the calibration of sets, the analysis of necessity relations, the construction of truth tables and the derivation of complex, parsimonious and intermediate solution types.

Calibration of sets

The process of translating base variables (also referred to as raw data) into condition or outcome set membership scores is called calibration, in fsQCA also fuzzification. In contrast to csQCA, continuous base variables need not be categorized directly in fsQCA but can be transformed with the help of continuous functions, a procedure called assignment by transformation (Verkuilen, 2005, p. 465). Ragin (2008), for example, suggests a piecewise-defined logistic function. Sufficient for the vast majority of fuzzification needs, QCA offers the `calibrateHI` function, one of whose flexible function classes for positive end-point concepts is given in Equation (1).

\[
\mu_S(b) = \begin{cases} 
0 & \text{if } \tau_{ex} \geq b, \\
\frac{1}{2} \left( \frac{\tau_{ex} - b}{\tau_{cr} - \tau_{in}} \right)^p & \text{if } \tau_{ex} < b \leq \tau_{cr}, \\
1 - \frac{1}{2} \left( \frac{\tau_{in} - b}{\tau_{in} - \tau_{cr}} \right)^q & \text{if } \tau_{cr} < b \leq \tau_{in}, \\
1 & \text{if } \tau_{in} < b. 
\end{cases}
\] (1)

Here, \( b \) is the base variable, \( \tau_{ex} \) the threshold for full exclusion from set \( S \), \( \tau_{cr} \) the crossover threshold at the point of maximally ambiguous membership in \( S \) and \( \tau_{in} \) the threshold for full inclusion in \( S \). The parameters \( p \) and \( q \) control the degrees of concentration and dilation. The piecewise-defined logistic membership function suggested in Ragin (2008) is also available. Furthermore, `calibrateHI` can generate set membership scores for sets based on negative or positive mid-point concepts (Thiem and Dusa, 2013, pp. 55-62). If no suitable thresholds have been found even after all means of external and internal identification have been exhausted, QCA’s `findThHI` function can be employed for searching thresholds using hierarchical cluster analysis.

\begin{verbatim}
> library(QCA)
> # base variable and vector of thresholds
> b <- sort(rnorm(15)); th <- quantile(b, c(0.1, 0.5, 0.9))

> # create bivalent crisp set
> calibrate(b, thresholds = th[2])
[1] 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1

> # create trivalent crisp set using thresholds derived from cluster analysis
> calibrate(b, thresholds = findTh(b, groups = 3))
[1] 0 0 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2

> # fuzzification using Equation (1)
> round(calibrate(b, type = "fuzzy", thresholds = th), 2)
[1] 0.00 0.00 0.04 0.32 0.42 0.47 0.48 0.50 0.59 0.72 0.77 0.93 0.94 1.00 1.00

> # negation of previous result
> round(calibrate(b, type = "fuzzy", thresholds = rev(th)), 2)
[1] 1.00 1.00 0.96 0.68 0.58 0.53 0.52 0.50 0.41 0.28 0.23 0.07 0.06 0.00 0.00

> # fuzzification using piecewise logistic
> round(calibrate(b, type = "fuzzy", thresholds = th, logistic = TRUE), 2)
[1] 0.02 0.04 0.06 0.25 0.38 0.45 0.46 0.50 0.64 0.79 0.83 0.93 0.93 0.96 0.97
\end{verbatim}

\textsuperscript{5}We prefer the term “inclusion” over the more common term “consistency” as introduced by Ragin (2006) because the measure represents the degree to which one set is included by another, producing evidence that is either consistent with the underlying hypothesis, inconsistent or mixed. The measure is also called Inclusion-1 index (cf. Smithson and Verkuilen, 2006, p. 65).
Analysis of necessity

Whenever the occurrence of an event B is accompanied by the occurrence of an event A, then B implies A \( B \Rightarrow A \) and A is implied by B \( A \Leftarrow B \). Put differently, A is necessary for B and B is sufficient for A. Transposed to the set-theoretic terminology of QCA, analyses of necessity proceed from the observation of a value under the outcome set \( Y \) - written \( Y \{ v_l \} \) - to the observation of a value under the condition set \( X \) - written \( X \{ v_l \} \). For analyzing necessity inclusion, the decisive question is to which degree objects are members of \( X \{ v_l \} \) and \( Y \{ v_l \} \) in relation to their overall membership in \( Y \{ v_l \} \).

If necessity inclusion is high enough, the evidence is consistent with the hypothesis that \( X \{ v_l \} \) is necessary for \( Y \{ v_l \} \) \((X \{ v_l \} \supseteq Y \{ v_l \})\). The formula for necessity inclusion \( \text{Incl}_N(X \{ v_l \}) \) is presented in Equation (2).

\[
\text{Incl}_N(X \{ v_l \}) = \frac{\sum_{i=1}^{n} \min(\{v_l\} x_i \{v_l\} y_i)}{\sum_{i=1}^{n} \{v_l\} y_i} \tag{2}
\]

Provided that \( X \{ v_l \} \supseteq Y \{ v_l \} \) is sufficiently true, necessity coverage allows an assessment of the frequency with which \( B \) occurs relative to \( A \). The formula for necessity coverage \( \text{Cov}_N(X \{ v_l \}) \) is given in Equation (3).

\[
\text{Cov}_N(X \{ v_l \}) = \frac{\sum_{i=1}^{n} \min(\{v_l\} x_i \{v_l\} y_i)}{\sum_{i=1}^{n} \{v_l\} x_i} \tag{3}
\]

For analyzing necessity relations, QCA offers the superSubset() function. If \( p_j \) denotes the number of values of condition set \( j \) with \( j = 1, 2, \ldots, k \), the function returns necessity inclusion, PRI and coverage scores for those of the \( d = \prod_{j=1}^{k} (p_j + 1) - 1 \) combinations of condition set values that just meet the given inclusion and coverage cut-offs.\(^6\) Therefore, superSubset() does not require users to predefine the combinations to be tested, and so removes the risk of leaving potentially interesting results undiscovered. The initial set of combinations always consists of all \( \prod_{j=1}^{k} p_j \) trivial intersections \( \langle x_1 \{v_l\}, x_2 \{v_l\}, \ldots, x_k \{v_l\} \rangle \). The size of the intersection is incrementally increased from 1 to k until its inclusion score falls below the cut-off. If no trivial intersection passes the inclusion cut-off, superSubset() automatically switches to forming set unions until the least complex form has been found.

For demonstration purposes, we reanalyze the data from Krook’s (2010) csQCA on women’s representation in 22 Western democratic parliaments. Countries with electoral systems of proportional representation (ES), parliamentary quotas (QU), social democratic welfare systems (WS), autonomous women’s movements (WM), more than 7% left party seats (LP) and more than 30% seats held by women (WNP) are coded “1”, all others “0”. The first five sets are the conditions to be tested for necessity in relation to the outcome set WNP. For reasons of simplicity and space, we use lower case letters for denoting set negation in all remaining code examples.

\[
\begin{align*}
\&> \text{data(Krook)} \\
\&> \text{Krook} \\
\&> \text{ES QU WS WM LP WNP} \\
\&> SE 1 1 1 0 0 1 \\
\&> FI 1 0 1 0 0 1 \\
\&> NO 1 1 1 1 1 1 \\
\&> \ldots \ldots \ldots \\
\&> <\text{rest omitted}> \\
\&> \text{superSubset(Krook, outcome = "WNP", cov.cut = 0.52)}
\end{align*}
\]

<table>
<thead>
<tr>
<th>incl</th>
<th>PRI</th>
<th>cov.r</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ES+LP</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>2 ES+QU</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>3 WS+WM+LP</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>4 QU+WM+LP</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>5 QU+WM+lp</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>6 QU+WS+LP</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>7 QU+WS+WM</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>8 es+QU+WS</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

\(^6\)The inclusion cut-off always enjoys priority over the coverage cut-off.
Whenever the occurrence of an event $A$ is accompanied by the occurrence of an event $B$, then $A$ implies $B$ ($A \Rightarrow B$) and $B$ is implied by $A$ ($B \Leftarrow A$). Put differently, $A$ is sufficient for $B$ and $B$ is necessary for $A$. Transposed to the set-theoretic terminology of QCA, analyses of sufficiency proceed from the observation of a value under $X$ to the observation of a value under $Y$. For analyzing sufficiency relations, the decisive question is to which degree objects are members of $X(v_i)$ and $Y(v_j)$ in relation to their overall membership in $X(v_j)$. If sufficiency inclusion is high enough, the evidence is consistent with the hypothesis that $X(v_i)$ is sufficient for $Y(v_j)$ ($X(v_i) \subseteq Y(v_j)$). The formula for sufficiency inclusion $\text{Incl}_S(X(v_i))$ is presented in Equation (4).

$$\text{Incl}_S(X(v_i)) = \frac{\sum_{i=1}^{n} \min((v_i)_x, (v_j)_y)}{\sum_{i=1}^{n} (v_i)_x}$$ (4)

The classical device for analyzing sufficiency relations is the truth table, which lists all $d = \prod_{i=1}^{k} p_i$ configurations and their corresponding outcome value. Configurations represent exhaustive combinations of set values characterizing the objects. For illustration, a simple hypothetical truth table with three bivalent condition sets $X_1, X_2$ and $X_3$ and the outcome value OUT is presented in Table 2. Three bivalent conditions yield the eight configurations listed under $C_i$. The minimum number of cases $n$ that is usually required for the respective outcome value is also appended.

<table>
<thead>
<tr>
<th>$C_i$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>OUT</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>≥1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>≥1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>≥1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>≥1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>≥1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>≥2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>0</td>
</tr>
</tbody>
</table>

It is important to emphasize that the outcome value is not the same as the outcome set, the latter of which does not show up in QCA truth table. Instead, the outcome value is based on the sufficiency inclusion score, returning a truth value that indicates the degree to which the evidence is consistent with the hypothesis that a sufficiency relation between a configuration and the outcome set exists. Configurations $C_1$ to $C_4$ are positive because they support this hypothesis (OUT = 1), $C_5$ is negative because it does not (OUT = 0). Mixed evidence exists for $C_6$ (OUT = C). If at least two objects conform to one configuration, but the evidence neither sufficiently confirms nor falsifies the hypothesis of a subset relation between this configuration and the outcome set, contradictions arise. No empirical evidence at all exists for $C_7$ and $C_8$. If a configuration has no or too few cases, it is called a logical remainder (OUT = ?).

The truthTable() function can generate truth tables for all three main QCA variants without users having to specify which variant they use. The structure of the data is automatically transposed into the correct format.

```r
> KrookTT <- truthTable(Krook, outcome = "WNP")
> KrookTT

OUT: outcome value
n: number of cases in configuration

7This table is also often called “table of combinations” (Mccluskey, 1965) or “function table” (Hohn, 1966).
At a minimum, truthTable() requires an appropriate dataset and the outcome argument, which identifies the outcome set. If conditions is not provided as an argument, it is assumed that all other sets in the data but the outcome are the conditions. By default, logical remainders are not printed unless specified otherwise by the logical argument complete. The logical argument show_cases prints the names of the objects next to the configuration in which they have membership above 0.5.

The truthTable() function includes three cut-off arguments that influence how OUT is coded. These are n.cut, incl.cut1 and incl.cut0. The first argument n.cut sets the minimum number of cases with membership above 0.5 needed in order to not code a configuration as a logical remainder. The second argument incl.cut1 specifies the minimal sufficiency inclusion score for a non-remainder configuration to be coded as positive. The third argument incl.cut0 offers the possibility of coding configurations as contradictions when their inclusion score is neither high enough to consider them as positive nor low enough to code them as negative. If the inclusion score of a non-remainder configuration falls below incl.cut0, this configuration is always considered negative. By means of the sort.by argument, the truth table can also be ordered along inclusion scores, numbers of cases or both, in increasing or decreasing order. If the original condition set labels are rather long, the logical letters argument can be used to replace the set labels with upper case letters in alphabetical order.

The leftmost column list the configuration row index values from the complete truth table. Sufficiency inclusion and PRI scores are also provided in the two rightmost columns. Once the truth table is fully coded, it can be minimized according to the theorems of Boolean algebra (McCluskey, 1965, pp. 84-89).

Analysis of sufficiency, step 2: Boolean minimization

The canonical union resulting from the truth table presented in Table 2 is given by Equation (5). It consists of four fundamental intersections (FI), each of which corresponds to one positive configuration. Generally, all FIs also represent positive configurations, but not all positive configurations become FIs. The analyst may decide to exclude some of these configurations from the minimization process on theoretical or empirical grounds.

If two FIs differ on the values of one condition only, then this condition can be eliminated so that a simpler term results. For example, Equation (5) can be reduced in two passes as shown in Figure 2. In the first pass, the four FIs can be reduced to four simpler terms. In the second pass, these four terms can then be reduced at once to a single term. No further reduction is possible, $X_1$ is the only term which is essential with respect to the outcome ($X_1 \subseteq Y$). All terms that survive the Boolean minimization process are called prime implicants (PI).

The central function of the QCA package that performs the minimization is eqmcc() (enhanced Quine-McCluskey) (Duša, 2007, 2010). It can derive complex, parsimonious and intermediate solutions from a truth table object or a suitable dataset. In contrast to complex solutions, parsimonious solutions incorporate logical remainders into the minimization process without any prior assessment by the analyst as to whether a sufficiency relation is plausible or not. Intermediate solutions offer a middle way insofar as those logical remainders that have been used in the derivation of the parsimonious...
solution are filtered according to the analyst’s directional expectations about the impact of each single condition set value on the overall sufficiency relation of the configuration of which it is part and the outcome set. By formulating such expectations, difficult logical remainders are excluded as FIs from the canonical union, whereas those logical remainders that enter the canonical union are easy. The complex solution, which is the default option, can be generated by eqmcc() with minimal typing effort.

> KrockSC <- eqmcc(KrockTT, details = TRUE)
> KrockSC

n OUT = 1/0/C: 11/11/0
  Total : 22

S1: ES*QU*ws*LP + ES*QU*ws*WM + es/ws*WM*LP + ES*WS*WM*LP + ES*WS*WM*LP

<table>
<thead>
<tr>
<th>incl PRI</th>
<th>cov.r</th>
<th>cov.u</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES<em>QU</em>ws*LP 1.000</td>
<td>1.000</td>
<td>0.273</td>
</tr>
<tr>
<td>ES<em>QU</em>ws*WM 1.000</td>
<td>1.000</td>
<td>0.273</td>
</tr>
<tr>
<td>es/ws<em>WM</em>LP 1.000</td>
<td>1.000</td>
<td>0.182</td>
</tr>
<tr>
<td>ES<em>WS</em>WM*LP 1.000</td>
<td>1.000</td>
<td>0.273</td>
</tr>
</tbody>
</table>

S1: 1.000 1.000 1.000

The truth table object KrockTT that was generated above is passed to eqmcc(). No further information is necessary in order to arrive at the complex solution. The logical argument details causes all parameters of fit to be printed together with the minimal union S1: inclusion (incl), PRI (PRI), raw coverage (cov.r) and unique coverage (cov.u) scores for each PI as well as the minimal union. If details = TRUE, the logical argument showNcases also prints the names of the objects that are covered by each PI.

If alternative minimal unions exist, all of them are printed if the row dominance principle for PIs is not applied as specified in the logical argument rowdom. One PI $P_1$ dominates another $P_2$ if all FIs covered by $P_2$ are also covered by $P_1$ and both are not interchangeable (cf. McCluskey, 1965, p. 150). Inessential PIs are listed in brackets in the solution output and at the end of the PI part in the parameters-of-fit table, together with their unique coverage scores under each individual minimal union. For example, the parsimonious solution without row dominance applied can be derived by making all logical remainders available for inclusion in the canonical union as FIs and by setting rowdom to FALSE.

> KrockSP <- eqmcc(KrockTT, include = "?", rowdom = FALSE, details = TRUE)
> KrockSP

n OUT = 1/0/C: 11/11/0
  Total : 22

S1: WS + ES*WM + QU*LP + (es*LP)
S2: WS + ES*WM + QU*LP + (WM*LP)

<table>
<thead>
<tr>
<th>incl PRI</th>
<th>cov.r</th>
<th>cov.u</th>
</tr>
</thead>
<tbody>
<tr>
<td>es/ws<em>WM</em>LP 1.000</td>
<td>1.000</td>
<td>0.273</td>
</tr>
</tbody>
</table>

8 Unique coverage scores do not apply to minimal unions.
The intermediate solution for bivalent set data requires a vector of directional expectations in the `direxp` argument, where “0” denotes absence, “1” presence and “-1” neither. The intermediate solution with all conditions expected to contribute to a positive outcome value when present is generated as follows:

```r
> KrookSI <- eqmcc(KrookTT, include = "?", direxp = c(1,1,1,1,1), details = TRUE)
> KrookSI

n OUT = 1/0/C: 11/11/0
Total : 22

p.sol: WS + ES+WM + QU+LP + WM+LP

S1: ES+WS + WM+LP + ES+QU+LP + ES+QU+WM

incl PRI cov.r cov.u

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ES+WS</td>
<td>1.00</td>
<td>1.00</td>
<td>0.455</td>
<td>0.182</td>
</tr>
<tr>
<td>WM+LP</td>
<td>1.00</td>
<td>1.00</td>
<td>0.545</td>
<td>0.091</td>
</tr>
<tr>
<td>ES+QU+LP</td>
<td>1.00</td>
<td>1.00</td>
<td>0.455</td>
<td>0.091</td>
</tr>
<tr>
<td>ES+QU+WM</td>
<td>1.00</td>
<td>1.00</td>
<td>0.545</td>
<td>0.091</td>
</tr>
</tbody>
</table>

S1: 1.00 1.00 1.00

For intermediate solutions, `eqmcc()` also prints the parsimonious solution (p.sol) whose simplifying assumptions have been used in filtering logical remainders. The PI chart of this intermediate solution (i.sol) that has been derived from the (first and only) complex and the (first and only) parsimonious solution (C1P1) can then be inspected by accessing the corresponding component in the returned object.

```r
> KrookSI$PIchart$i.sol$C1P1
```

```
  4  12  21  24  26  27  28  29  32
ES+WS - - x - - - x - x
WM+LP - - - x - - - x - x
ES+QU+LP - - - x - - - x - x
ES+QU+WM - - - - - x - x - x
```

If several minimal sums exist under both the parsimonious and complex solution, the PI chart of the respective combination for the intermediate solution can be accessed by replacing the numbers in the C1P1 component.

Besides the PI chart, the solution object returned by `eqmcc()` also contains a dataframe of PI set membership scores in the `pims` component. These scores can then be used to draw Venn diagrams of solutions, similar to the one shown in Figure 3, using suitable R packages such as `VennDiagram` (Chen and Boutros, 2011).

```r
> KrookSI$pims$i.sol$C1P1
```

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ES+WS</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>WM+LP</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>ES+QU+LP</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>ES+QU+WM</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>
```
Figure 3: Venn diagram of three PIs from intermediate solution produced with the VennDiagram package.

Summary

In recent years, Qualitative Comparative Analysis (QCA) has become the method of choice for testing configurational hypotheses. However, QCA is still very much a “method in the making”. Extensions, enhancements and alternative algorithms appear on a regular basis (Baumgartner, 2009; Eliason and Stryker, 2009; Schneider and Wagemann, 2012; Thiem, 2012). R provides an ideal environment within which established QCA procedures as well as more advanced techniques can be implemented in a manner as transparent and user-responsive as possible. The QCA package makes a significant contribution in this regard. It fills the individual gaps in other programs’ coverage of basic functionality, and provides further improvements through complementary and advanced procedures.

The QCA software market remains dominated by the two graphical user interface programmes fs/QCA and Tosmana. QCA seeks to bridge the method of QCA with powerful command-line software while retaining a user-friendly code and command structure. In order to lower the barriers for social scientists to choosing R for QCA further, we have published an introductory textbook with extended replications of recent QCA studies from various research areas (Thiem and Duşa, 2013). Although most examples have been taken from political science, the book may also be of interest to researchers from related disciples. At the same time, this textbook also serves as a comprehensive reference manual for the QCA package.

Bibliography


L. Cronqvist. Tosmana, version 1.3.2.0 [computer program]. University of Trier, Trier, 2011. [p88]


R. Huang. QCA3: Yet another package for qualitative comparative analysis, 2011. R package version 0.0-4. [p88]


C. C. Ragin and S. Davey. *fs/QCA, version 2.5 [computer program]*. Department of Sociology, University of Arizona, Tucson, AZ, 2009. [p88]

C. Reichert and C. Rubinson. *Kirq, version 2.1.9 [computer program]*. University of Houston-Downtown, Houston, TX, 2013. [p88]


Alrik Thiem
Department of Humanities, Social and Political Sciences
Swiss Federal Institute of Technology Zurich (ETHZ)
IFW C29.2, Haldeneggsteig 4
8092 Zurich
Switzerland
thiem@sipo.gess.ethz.ch

Adrian Duşa
Department of Sociology
University of Bucharest
1 Schitu Măgureanu
050025 Bucharest
Romania
dusadrian@unibuc.ro