Qualitative Comparative Analysis using QCA3

Ronggui Huang

April 18, 2014

1 Introduction

QCA3 can do various types of qualitative comparative analysis, namely crisp set QCA, fuzzy set QCA and multi-value QCA. It allows inclusion of simplifying assumptions and can produce intermediate solutions.

All examples in Rihoux and Ragin (2009) can be reproduced by QCA3 package. To reproduce the following examples, you need to load the QCA3 package first by

```
> library(QCA3)
```

2 Crisp set QCA

Before conducting crisp set QCA (csQCA) with QCA3, you need to import your dataset into R. If your data is in Excel format, I would recommend you to export the data to a csv file and import the csv file into R by read.csv function. In this article, all datasets are shipped with the QCA3 package, thus I will skip the steps of data import.

In crisp set QCA (csQCA), all variables are binary (0 or 1). The dataset in this example is Lipset_cs. The first step is to construct a truth table. You can use the function of cs_truthTable to do it.

```
> (cst <- cs_truthTable(Lipset_cs,outcome="SURVIVAL",
+                       condition=c("GNPCAP", "URBANIZA", "LITERACY", "INDLAB", "GOVSTAB"),
+                       cases="CASEID"))

configuration distribution
  0   1 Sum
  6   3   9
case distribution
 OUT NCase
   0    10
   1     8
=====
    GNPCAP URBANIZA LITERACY INDLAB GOVSTAB NCase freq1 freq0 OUT
122      0        0        0      0       0     3     0     3   0
131      0        0        1      0       0     2     0     2   0
159      1        0        1      1       0     1     0     1   0
162      1        1        1      1       0     1     0     1   0
203      0        0        0      0       1     2     0     2   0
212      0        0        1      0       1     1     0     1   0
213      1        0        1      0       1     2     2     0   1
240      1        0        1      1       1     2     2     0   1
243      1        1        1      1       1     4     4     0   1
               Cases
```

```
122     GRE, POR, SPA
131          HUN, POL
159               AUS
162               GER
203          ITA, ROM
212               EST
213          FIN, IRE
240          FRA, SWE
243 BEL, CZE, NET, UK
```

The above command constructs a truth table and assigns it to an object called cst, which can be used in the following analyses. You can choose any legitimate object name to store the produced truth table as long as it is a legitimate name in R.

In this command, Lipset_cs is the data frame which contains all the variables. The argument of outcome specificifies the explained variable, say "SURVIVAL" in this example (note that you need to enclose SURVIVAL with quotation mark). The argument of condition specificies the explanatory variables. It is a string vector, each element of which is a condition or explanatory variable. At least two conditions are needed. In this example, five conditions are provided. All variables, be outcome or conditions, are in the data frame of Lipset_cs. For more details, you can refer to the help page of cs_truthTable.

The truth table is constructed by cs_truthTable in an automatic manner. However, you can also manually override the outcome by function of setOUT, but I will not go through the details here.

At this moment, you need to examine the truth table closely. Perhaps, it is a good idea to make connections between the truth table and the detailed empirical evidence about each case. Special attention should be paid to contradictory configurations, and you need to to handle such contradictory configurations if any. Once you have a truth table without contradictory configuration or you have come up a strategy to handle them, you can move to the next step, minimization of the truth table without remainders. All you need to d is to pass the truth table, cst, produced previously, to the reduce function.

```
> reduce(cst)

Call:
reduce(x = cst)


----------------
Explaining 3 configuration(s)

----------------
Prime implicant No. 1 with 2 implicant(s)

GNPCAP*urbaniza*LITERACY*GOVSTAB + GNPCAP*LITERACY*INDLAB*GOVSTAB

Common configuration: GNPCAP*LITERACY*GOVSTAB
```

The default explains positive outcome (in this case, SURVIVAL=1). If you want to explain negative outcome, you need to set the argument of explain to "negative", which is the third step.

```
> reduce(cst, explain="negative")

Call:
reduce(x = cst, explain = "negative")
```

```
----------------
Explaining 6 configuration(s)

----------------
Prime implicant No. 1 with 2 implicant(s)

gnpcap*urbaniza*indlab + GNPCAP*LITERACY*INDLAB*govstab

Common configuration: None
```

By default, remainders are not used and no simplifying assumption is made. The fourth step is to get the most parsimonious solution to positive outcome by including remainders. All you need to do is to set the argument of remainders to "include".

```
> reduce(cst, remainders="include")

Call:
reduce(x = cst, remainders = "include")


----------------
Explaining 3 configuration(s)

----------------
Prime implicant No. 1 with 1 implicant(s)

GNPCAP*GOVSTAB

Common configuration: GNPCAP*GOVSTAB
```

Similarly, you can explain negative outcome by including remainders. Now, you need to specify both arguments of explain and remainders.

```
> reduce(cst, explain="negative", remainders="include")

Call:
reduce(x = cst, explain = "negative", remainders = "include")


----------------
Explaining 6 configuration(s)

----------------
Prime implicant No. 1 with 2 implicant(s)

gnpcap + govstab

Common configuration: None
```

Now, you may wonder what remainders have been included. It is always a good idea to examine them. To do so, you need to assign the return of reduce to an object first. Take the explanation of negative outcome for example. Let assign it to an object called ansNeg. Then you can pass ansNeg to the function of SA, which will return a list of remainders used in the minimization (which are also called simplifying assuptions). It shows that 18 remainders have been included.

```
> ansNeg <- reduce(cst, explain="negative", remainders="include")
> SA(ansNeg)

Simplifying Assumptions

----------------
Prime implicant No. 1 with 18 implicant(s)

gnpcap*URBANIZA*literacy*indlab*govstab +
gnpcap*URBANIZA*LITERACY*indlab*govstab +
gnpcap*urbaniza*literacy*INDLAB*govstab +
gnpcap*URBANIZA*literacy*INDLAB*govstab +
gnpcap*urbaniza*LITERACY*INDLAB*govstab +
gnpcap*URBANIZA*LITERACY*INDLAB*govstab +
gnpcap*URBANIZA*literacy*indlab*GOVSTAB +
gnpcap*URBANIZA*LITERACY*indlab*GOVSTAB +
gnpcap*urbaniza*literacy*INDLAB*GOVSTAB +
gnpcap*URBANIZA*literacy*INDLAB*GOVSTAB +
gnpcap*urbaniza*LITERACY*INDLAB*GOVSTAB +
gnpcap*URBANIZA*LITERACY*INDLAB*GOVSTAB +
GNPCAP*urbaniza*literacy*indlab*govstab +
GNPCAP*URBANIZA*literacy*indlab*govstab +
GNPCAP*urbaniza*LITERACY*indlab*govstab +
GNPCAP*URBANIZA*LITERACY*indlab*govstab +
GNPCAP*urbaniza*literacy*INDLAB*govstab +
GNPCAP*URBANIZA*literacy*INDLAB*govstab

Common configuration: None
```

3 fuzzy set QCA

3.1 Calibration

Fuzzy set QCA (fsQCA) requires fuzzy set scores, which range from 0 to 1. The fuzzy membership scores can be directly assigned according to substantive knowledge about the cases at hand.

Often, researchers have interval measures and need to convert such measures into fuzzy membership scores. This process is called caliberation, which is described in Ragin (2008). The QCA3 package implements the direct method of caliberation. This method requires researchers to choose three anchors, namely fullin, fullout and crossover values. The fullin value is recoded as fuzzy set score of 0.953. The fullout value is recoded as fuzzy set score of 0.03, and the crossover value is recoded as fuzzy set score of 0.5 [1]. Researchers must choose the three anchors based on substantive knowledge. One you have the fullin, fullout and corssover values, you can caliberate the fuzzy set scores by the directCaliberate function.

For example, the data frame of Lipset_fs has an interval variable of Developed, and the following command caliberate it with three anchors of 900 (fullin), 400 (fullout), and 500 (crossover).

```
> directCalibration(Lipset_fs$Developed,fullin=900,fullout=400, crossover=550)

 [1] 0.811798852 0.991092956 0.576774248 0.161762067 0.585147064 0.976411872
 [7] 0.891545632 0.038787464 0.073921688 0.723727959 0.340271893 0.980887270
[13] 0.017764604 0.009809787 0.012202676 0.024806018 0.951831001 0.985168469
```

[1] 0.953 and 0.03 are chosen because the log odds of them are 3 and -3 respectively, which are approximate to the description in Ragin (2008). However, users should note that the result is slightly different from fs/QCA because exact log odds of the fullin and fullout fuzzy membership scores rather than 3 and -3 are used in the QCA3 package.

You can also add the returned fuzzy set score to the data frame (Lipset_fs) as a variable (DFZ) by,

```
> Lipset_fs$DFZ <- directCalibration(Lipset_fs$Developed,900,400, 550)
```

3.2 Exploratory Analysis

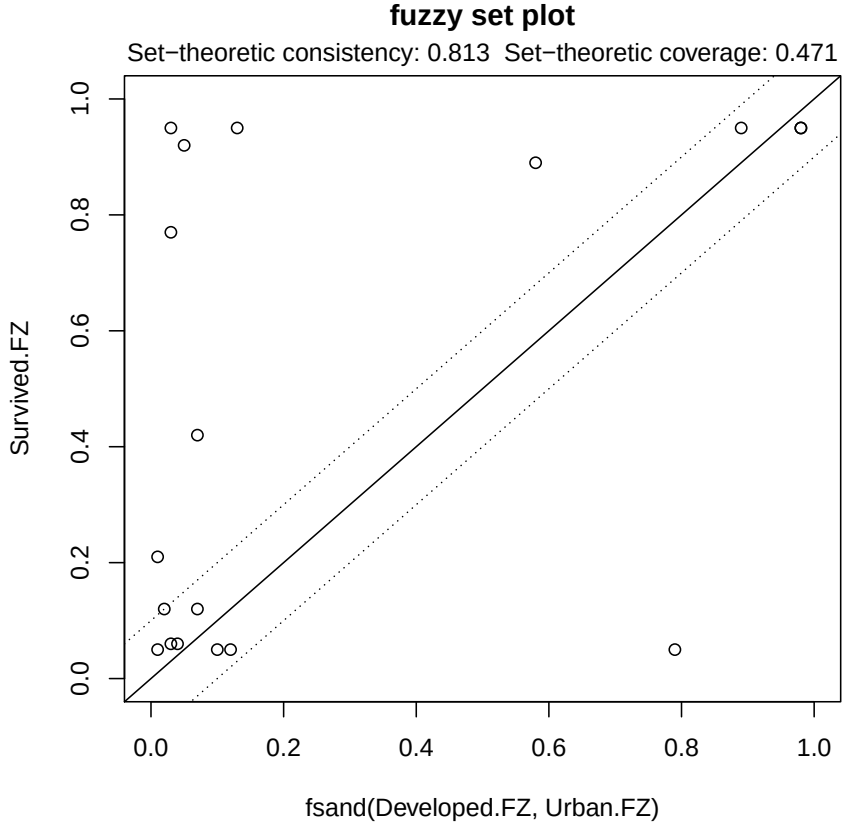
Before jumping into fuzzy set QCA, it is always a good idea to explore the bivariate relationship between outcome and condition variables. The key measures are sufficiency scores and necessary scores. The suffnec function takes a data frame as input and produces both sufficiency scores and necessary scores matrices.

```
> suffnec(Lipset_fs[,c("Survived.FZ","Developed.FZ","Urban.FZ",
+                      "Literate.FZ","Industrial.FZ", "Stable.FZ")])

Necessity Scores Matrix:
'X is necessary condition of Y'
               Y
X               Survived.FZ Developed.FZ Urban.FZ Literate.FZ Industrial.FZ
  Survived.FZ         1.000        0.775    0.771       0.643         0.684
  Developed.FZ        0.831        1.000    0.829       0.695         0.862
  Urban.FZ            0.539        0.539    1.000       0.436         0.640
  Literate.FZ         0.991        0.999    0.961       1.000         0.935
  Industrial.FZ       0.669        0.786    0.896       0.593         1.000
  Stable.FZ           0.920        0.884    0.852       0.749         0.868
               Y
X               Stable.FZ
  Survived.FZ       0.707
  Developed.FZ      0.729
  Urban.FZ          0.457
  Literate.FZ       0.887
  Industrial.FZ     0.652
  Stable.FZ         1.000

Sufficiency Scores Matrix:
'X is sufficient condition of Y'
               Y
X               Survived.FZ Developed.FZ Urban.FZ Literate.FZ Industrial.FZ
  Survived.FZ         1.000        0.831    0.539       0.991         0.669
  Developed.FZ        0.775        1.000    0.539       0.999         0.786
  Urban.FZ            0.771        0.829    1.000       0.961         0.896
  Literate.FZ         0.643        0.695    0.436       1.000         0.593
  Industrial.FZ       0.684        0.862    0.640       0.935         1.000
  Stable.FZ           0.707        0.729    0.457       0.887         0.652
               Y
X               Stable.FZ
  Survived.FZ       0.920
  Developed.FZ      0.884
  Urban.FZ          0.852
  Literate.FZ       0.749
  Industrial.FZ     0.868
  Stable.FZ         1.000
```

In the above command, Lipset_fs is the name of a data frame. Here I use the "[" operator to extract a relevant subset, which is then passed to suffnec.

You can use a graph to depict the set-theoretic consistency and coverage. The function fsplot provides a friendly interface. For example, if you want to examine whether development and urbanization are sufficient for regime survival, you can use the following command.

```
> fsplot(Survived.FZ~fsand(Developed.FZ, Urban.FZ),data=Lipset_fs)
```



The fsplot use a formula interface. The left hand side variable is the outcome variable. The right hand side is explanatory variable. When the explanatory variable is a conjunctural condition, you can use fsand to get the conjunctural condition before plotting. In the above example, the right hand side is fsand(Developed.FZ, Urban.FZ), which suggests that Developed.FZ*Urban.FZ is regarded as a conjunctural condition.

The consistency and coverage scores are displayed in the figure, which eases the interpretation of the result.

3.3 Fuzzy set truth table and minimization

At this point, we have made up our minds concerning the conditions, and we can construct a fuzzy set truth table for further analysis. In the textbook example, five conditions are chosen, and the consistency threshold is set to 0.7.

```
> conditions <- c("Developed.FZ","Urban.FZ","Literate.FZ","Industrial.FZ", "Stable.FZ")
> fst <- fs_truthTable(Lipset_fs,"Survived.FZ", conditions, consistency=0.7)
> print(fst)

configuration distribution
  0   1 Sum
```

```
  6   3   9
case distribution
 OUT NCase
   0    10
   1     8
=====
    Developed.FZ Urban.FZ Literate.FZ Industrial.FZ Stable.FZ freq1 freq0 NCase
243            1        1           1             1         1     4     0     4
240            1        0           1             1         1     2     0     2
213            1        0           1             0         1     2     0     2
212            0        0           1             0         1     0     1     1
203            0        0           0             0         1     0     2     2
162            1        1           1             1         0     0     1     1
159            1        0           1             1         0     0     1     1
131            0        0           1             0         0     0     2     2
122            0        0           0             0         0     0     3     3
    OUT Consistency priConsistency sqrtProduct      Cases
243   1   0.9042056     0.88579387  0.80093979 2,3,12,18/
240   1   0.7087719     0.63436123  0.44961744      6,17/
213   1   0.8042705     0.71938776  0.57858232      5,10/
212   0   0.5285714     0.22807018  0.12055138         /4
203   0   0.2780269     0.00000000  0.00000000     /11,15
162   0   0.4452555     0.05000000  0.02226277         /7
159   0   0.3782051     0.03960396  0.01497842         /1
131   0   0.5209003     0.11309524  0.05891135      /9,13
122   0   0.2159763     0.00000000  0.00000000   /8,14,16
```

The above commands construct a fuzzy set truth table, assign it to an object called fst, and then print it. Next, we simplify the truth table through Boolean minimization.

```
> fsans <- reduce(fst)
> print(fsans)

Call:
reduce(x = fst)


----------------
Explaining 3 configuration(s)

----------------
Prime implicant No. 1 with 2 implicant(s)

DEVELOPED.FZ*urban.fz*LITERATE.FZ*STABLE.FZ +
DEVELOPED.FZ*LITERATE.FZ*INDUSTRIAL.FZ*STABLE.FZ

Common configuration: DEVELOPED.FZ*LITERATE.FZ*STABLE.FZ
```

The print method only shows minimal information, but the summary method shows futher information on the goodness of fit.

```
> summary(fsans)

Call:
reduce(x = fst)
```

```
Total number of cases: 18
Number of cases [1]: 8
Number of cases [0]: 10
Explaining [1]

----------------
Prime implicant No. 1 with 2 implicant(s)

DEVELOPED.FZ*urban.fz*LITERATE.FZ*STABLE.FZ +
DEVELOPED.FZ*LITERATE.FZ*INDUSTRIAL.FZ*STABLE.FZ

Goodness of fit
                                                 consistency rawCoverage
DEVELOPED.FZ*urban.fz*LITERATE.FZ*STABLE.FZ        0.8092105   0.4330986
DEVELOPED.FZ*LITERATE.FZ*INDUSTRIAL.FZ*STABLE.FZ   0.8426073   0.6220657
[solution]                                         0.8712500   0.8180751
                                                 uniqueCoverage
DEVELOPED.FZ*urban.fz*LITERATE.FZ*STABLE.FZ           0.4330986
DEVELOPED.FZ*LITERATE.FZ*INDUSTRIAL.FZ*STABLE.FZ      0.0000000
[solution]                                            0.8180751

Number of cases: 4 + 6
Percentage of explained cases: 50% + 75%
Cases covered by multiple PIs: 2 (25%)
Cases: (2)6,17/ (1)5,10/ + (1)2,3,12,18/ (2)6,17/
```

There is an update method for QCA object, so you can use the following command to get a result including remainders in the minimization process.

```
> update(fsans, remainders="include")

Call:
reduce(x = fst, remainders = "include")


----------------
Explaining 3 configuration(s)

----------------
Prime implicant No. 1 with 1 implicant(s)

DEVELOPED.FZ*STABLE.FZ

Common configuration: DEVELOPED.FZ*STABLE.FZ
```

In fsQCA, it is always a good idea to generate a new variable indicating the fuzzy set membership in the negation set using fsnot. Using this new fuzzy set score to construct a new fuzzy set truth table and minimize it.

```
> Lipset_fs$Not.Survived <- fsnot(Lipset_fs$Survived.FZ)
> fst2 <- fs_truthTable(Lipset_fs,"Not.Survived", conditions, consistency=0.7)
> print(fst2)

configuration distribution
  0   1   C Sum
```

```
  3   5   1   9
case distribution
 OUT NCase
   0     8
   1     8
   C     2
=====
    Developed.FZ Urban.FZ Literate.FZ Industrial.FZ Stable.FZ freq1 freq0 NCase
243            1        1           1             1         1     0     4     4
240            1        0           1             1         1     0     2     2
213            1        0           1             0         1     0     2     2
212            0        0           1             0         1     1     0     1
203            0        0           0             0         1     2     0     2
162            1        1           1             1         0     1     0     1
159            1        0           1             1         0     1     0     1
131            0        0           1             0         0     1     1     2
122            0        0           0             0         0     3     0     3
    OUT Consistency priConsistency sqrtProduct      Cases
243   0   0.2500000      0.1058496   0.0264624 /2,3,12,18
240   0   0.4947368      0.3656388   0.1808950      /6,17
213   0   0.4982206      0.2806122   0.1398068      /5,10
212   1   0.8607143      0.7719298   0.6644110         4/
203   1   0.9820628      0.9751553   0.9576637     11,15/
162   1   0.9708029      0.9500000   0.9222628         7/
159   1   0.9743590      0.9603960   0.9357705         1/
131   C   0.8553055      0.7321429   0.6262058       13/9
122   1   1.0000000      1.0000000   1.0000000   8,14,16/

> fsans2 <- reduce(fst2)
> summary(fsans2)

Call:
reduce(x = fst2)

Total number of cases: 18
Number of cases [1]: 9
Number of cases [0]: 9
Explaining [1]

----------------
Prime implicant No. 1 with 3 implicant(s)

developed.fz*urban.fz*literate.fz*industrial.fz +
DEVELOPED.FZ*LITERATE.FZ*INDUSTRIAL.FZ*stable.fz +
developed.fz*urban.fz*industrial.fz*STABLE.FZ

Goodness of fit
                                                 consistency rawCoverage
developed.fz*urban.fz*literate.fz*industrial.fz    0.9905437   0.4419831
DEVELOPED.FZ*LITERATE.FZ*INDUSTRIAL.FZ*stable.fz   0.9812207   0.2204641
developed.fz*urban.fz*industrial.fz*STABLE.FZ      0.8900000   0.3755274
[solution]                                         0.9335180   0.7109705
                                                 uniqueCoverage
developed.fz*urban.fz*literate.fz*industrial.fz       0.4419831
```

```
DEVELOPED.FZ*LITERATE.FZ*INDUSTRIAL.FZ*stable.fz      0.0000000
developed.fz*urban.fz*industrial.fz*STABLE.FZ         0.0000000
[solution]                                            0.7109705

Number of cases: 5 + 2 + 3
Percentage of explained cases: 55.556% + 22.222% + 33.333%
Cases covered by multiple PIs: 2 (22.222%)
Cases: (2)11,15/ (1)8,14,16/ + (1)7/ (1)1/ + (1)4/ (2)11,15/
```

4 Session Information

```
> sessionInfo()

R Under development (unstable) (2014-04-17 r65403)
Platform: i386-w64-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=Chinese (Simplified)_People's Republic of China.936
[3] LC_MONETARY=Chinese (Simplified)_People's Republic of China.936
[4] LC_NUMERIC=C
[5] LC_TIME=Chinese (Simplified)_People's Republic of China.936

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] QCA3_0.0-6

loaded via a namespace (and not attached):
[1] lpSolveAPI_5.5.2.0-9 tools_3.2.0
```

References

C. C. Ragin. *Redesigning social inquiry: fuzzy sets and beyond.* Chicago: University of Chicago Press, 2008.

B. Rihoux and C. C. Ragin, editors. *Configurational comparative methods: qualitative comparative analysis (QCA) and related techniques.* Sage, Thousand Oaks, Aug 2009.