Qualitative Comparative Analysis with QCA3

Ronggui Huang

Oct. 2012

1 Introduction

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

All examples in Rihoux and Ragin (2009) can be fully reproduced by QCA3 package.

2 Crisp set QCA

Before you conduct 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, and 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 and examine it. 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"))

    GNPCAP URBANIZA LITERACY INDLAB GOVSTAB NCase freq1 freq0 OUT             Cases
122      0        0        0      0       0     3     0     3   0     GRE, POR, SPA
131      0        0        1      0       0     2     0     2   0          HUN, POL
159      1        0        1      1       0     1     0     1   0               AUS
162      1        1        1      1       0     1     0     1   0               GER
203      0        0        0      0       1     2     0     2   0          ITA, ROM
212      0        0        1      0       1     1     0     1   0               EST
213      1        0        1      0       1     2     2     0   1          FIN, IRE
240      1        0        1      1       1     2     2     0   1          FRA, SWE
243      1        1        1      1       1     4     4     0   1 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 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. At least two conditions are needed. In this example, five conditions are provided. All variables, be outcome or condition, 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 automatically. 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 to handle 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 including remainders.

All you need to to 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". That is our 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 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 it. 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 are 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
```

References

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