RISQ Representativity Indicators for Survey Quality

R-Cockpit manual

A Tool for Graphical Analysis of Representativity

Work package 8
Deliverable 12.2

Jelke Bethlehem
Centraal Bureau voor de Statistiek, The Netherlands

March 21, 2010

1. Introduction

The program *Cockpit* is a tool to conduct a graphical analysis of the representativity of the survey response. It shows in various ways how representativity is related to auxiliary variables.

The program was developed as a demonstration tool that shows some of the possibilities for graphical analysis of representativity. Sections 3 to 7 describe and show graphs that can be generated by the program. It is likely that use of representativity indicators, either to monitor the field work or to analyse the collected data, requires more extensive and/or tailor-made software.

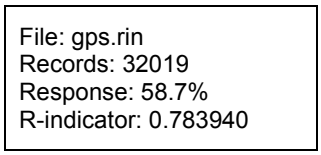
This program does not compute response probabilities. It is assumed that an input file has been created by SPSS, Stata, SAS, an R program or another software tool. This input file must contain the values of auxiliary variables and the estimated response probabilities. A description of the required input files is given in section 8. Other software tools for the R-indicator can be found on the website of the RISQ-project: www.risq-project.eu.

The GPS survey data set has been used in this document to illustrate the possible output of the program. This data set is based on a Dutch survey that has been carried by Statistics Netherlands. To avoid the disclosure of sensitive individual information, the data set has been anonymised. The sample size was a little over 32,000 persons. The response rate was almost 60%.

2. Starting the program

The program is simply started by double-clicking its name or icon. The first step is to read a data set. This is accomplished by clicking on the **Read file** button and selecting a file. Such a file must always have the extension *rin*. Section 8 gives more information about the contents of such a file. After reading the data, the program will display some general information about the file. Figure 2.1 gives an example.

Figure 2.1. General information about the data file



The data file contains 32019 records. This is the sample size. The response rate is equal to 58.7%. This quantity is usually equal to average response probability. The value of the R-indicator is equal to 0.783940. Since this value is lower than 1, there is some lack of representativity.

Suppose, the (estimated) response probabilities for the n elements in the sample are denoted by ρ_1, ρ_2, …, ρ_n. Then the R-indicator is computed as

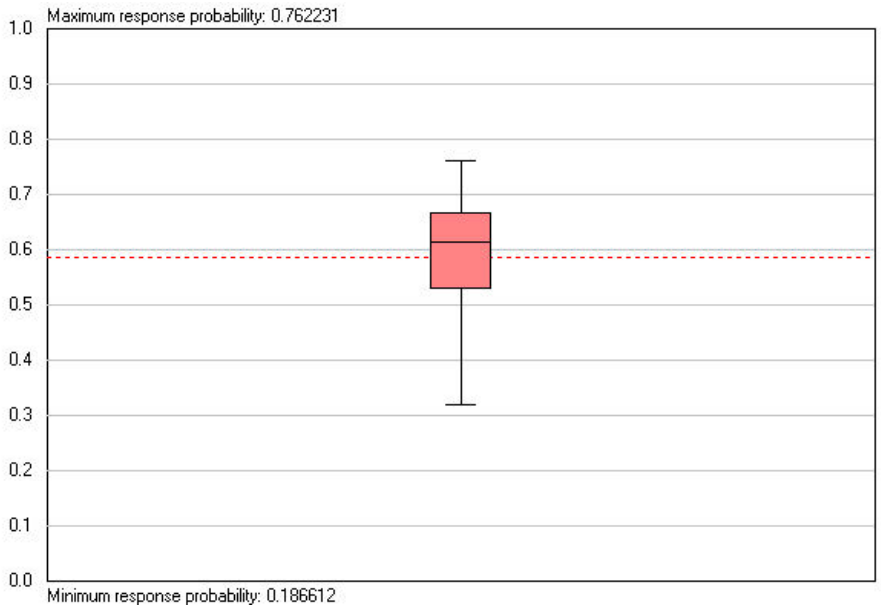
$$R = 1 - 2S(\rho) = 1 - 2\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\rho_i - \bar{\rho})^2} \qquad (1.1)$$

Note that no bias correction has been applied for the computation.

3. The overall box plot

The overall box plot shows the variation of the values of the (estimated) response probabilities in the input file. The box plot is automatically produced after reading the input file. Figure 3.1. contains an example.

Figure 3.1. An overall box plot



The red box represents the values between the first and third quartile. It is the middle half of the response probabilities. According to figure 3.1 the middle 50% of the response probabilities are approximately between 0.55 and 0.65. The black line in the box represents the median. The lines stretching out from both sides of the box extend to values that are just not considered outliers. The are the outmost values lying within a distance of 1.5 times the length of the box from the box. So, the 'regular' part of the distribution covers an interval from approximately 0.32 to 0.76. This shows there is considerable variation in response probabilities.

The horizontal red dashed line represents the mean of the response probabilities. If median and mean are different, this may be seen as an indication that there are substantial outliers.

The minimum value of the response probability (0.186612) is given below the plot, and the maximum value (0.762231) above the plot.

A plot like this can always be generated by the program by selecting *Boxplot* as the graph type and selecting no variable.

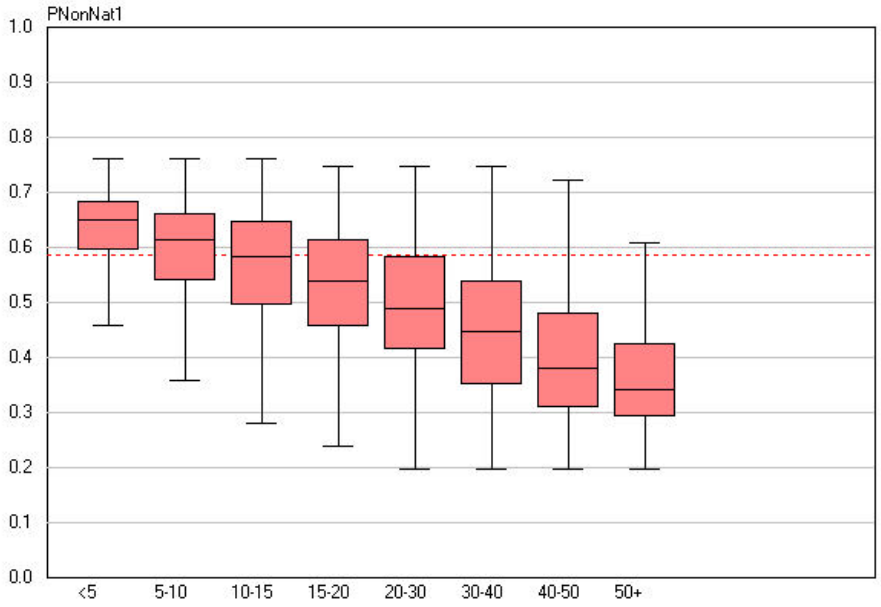
4. Box plots for the categories of a variable

For the analysis and correction of nonresponse it is important to find variables that are strongly related to response behaviour. The strongest form of relation is that in which the response probabilities only vary between the categories of a variable and not within the categories. There is no relationship between a variable and response behaviour if the response probabilities only vary within the categories of the variable and not between its categories.

The relationship between a variable and response behaviour can be explored by drawing box plots of the response probabilities for each category of the variable. Figure 4.1 shows an example of such a plot for the variable *PNonNat1*. This is the percentage of non-natives living in the neighbourhood (in 8 categories).

Figure 4.1. shows that the response probabilities vary both between and within the categories of *PNonNat*. The between category response probabilities show a clear pattern: the response probabilities decrease as the percentage of non-natives increases. This is an indication there is relationship between *PNonNat1* and the response behaviour. Therefore, this variable should, for example, be considered for inclusion in a weighting adjustment model.

Figure 4.1. Box plots for the percentage of non-natives in the neighbourhood



A plot like this can always be generated by the program by selecting *Boxplot* as the graph type and selecting a variable from the list.

5. Unconditional partial indicators at the variable level

The partial R-indicator measures how large the variation of the response probabilities between the categories of a variable is. The larger the between-category variation is, the stronger the relationship is.

Suppose a variable X has L categories. Let n_h denote the sample size in category h, for $h = 1, 2, .., L$. Then $n_1 + n_2 + \ldots + n_L = n$. Furthermore, let $\bar{\rho}$ be the mean response probability in the sample, and let $\bar{\rho}_h$ the mean of the response probabilities in category h of X.

The unconditional partial indicator for variable X is now defined as

$$P_U(X) = \sqrt{\frac{1}{n}\sum_{h=1}^{L} n_h(\bar{\rho}_h - \bar{\rho})^2} \quad . \tag{5.1}$$

Note that $P_U(X) \leq S(\rho) \leq 0.5$. i.e. the total variation between categories is smaller than the total variation.

By computing and comparing the unconditional partial indicators for a set of variables it can be established for which variables the relationships are strongest.

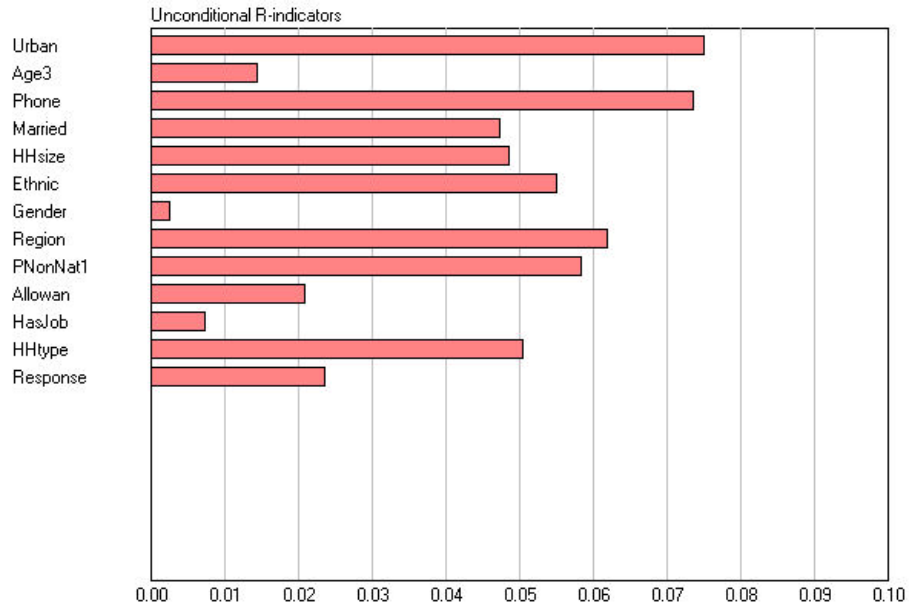
Figure 5.1 shows an example of the plot containing the unconditional partial indicators for all variables in the GSP data set. The values of the indicators may vary between 0 and 1. Note that the scale of the horizontal axis is adapted to make the differences between the indicators of variables clearer.

Apparently, the variables *Urban* (degree of urbanization) and *Phone* (has a listed phone number) show the strongest relationship with response behaviour. The relation is weakest for *Gender* and *Age3* (age in three categories).

A plot like this can be generated by the program by selecting *Unconditional R-indicator* as the graph type, and selecting no variable form the list of variables.

.

Figure 5.1. Unconditional partial indicators



6. Unconditional partial indicators at the category level

The partial R-indicator can give more information about the relationship of a variable X and response behaviour if this indicator is compute for each category of X separately. It is clear from expression 5.1 that each category h contributes an amount

$$\frac{n_h}{n}\left(\bar{\rho}_h - \bar{\rho}\right)^2 \tag{6.1}$$

to $P_U(X)$. The unconditional partial indicators within categories are obtained by taking the square root of the quantities in (6.1). The unconditional partial indicator for category h is denoted by

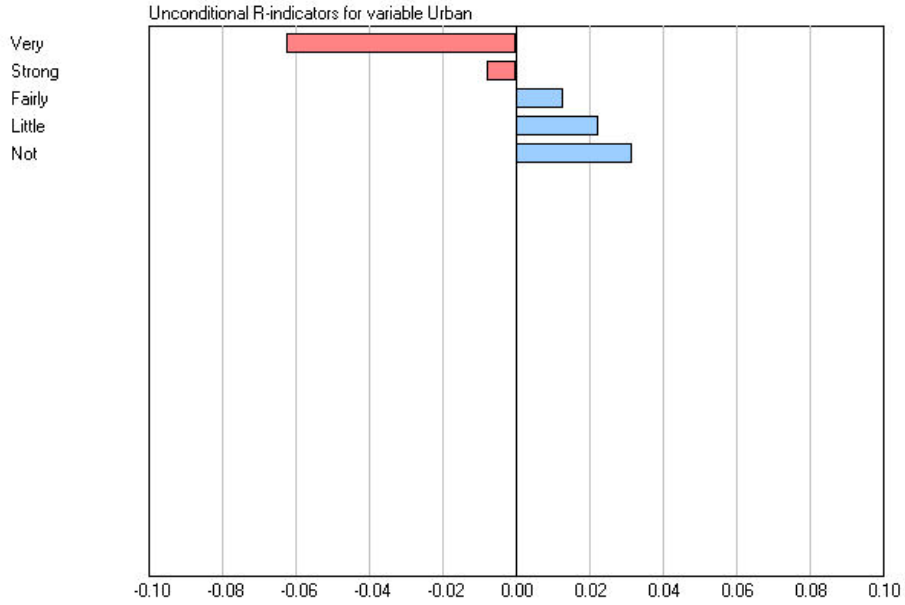
$$P_U(X,h) = \sqrt{\frac{n_h}{n}}\left(\bar{\rho}_h - \bar{\rho}\right). \tag{6.2}$$

$P_U(X, h)$ can assume positive and negative values. A positive value means that the particular category is over-represented. A negative value means that the particular category is under-represented.

Figure 6.1 shows the unconditional indicators for the categories of the variable degree of urbanisation in the GPS data set. Negative values are presented by red bars and positive values by blue bars. A plot like this can be generated by the program by selecting *Unconditional R-indicator* as the graph type and selecting a variable from the list.

It is clear from the plot that particularly people from very strongly urbanised areas (the big cities) are under-represented. The response probabilities are higher than average in the rural (not urbanised) areas.

Figure 6.1. Unconditional partial indicators by category



7. Conditional partial indicators at the variable level

Conditional partial indicators can only be computed for variables that are included in the model for estimating the response probabilities. These indicators measure the relative importance of these variables.

The conditional partial indicator for a variable X is obtained by cross-classification all model variables, but with the exception of X. Suppose, this cross-classification results in L cells U_1, U_2, …, U_L. Let n_h denote the sample size in cell h, for $h = 1, 2, .., L$. Then $n_1 + n_2 + \ldots + n_L = n$. Furthermore, let $\bar{\rho}_h$ the mean of the response probabilities in cell h.

The conditional partial indicator for variable X is now defined as

$$P_C(X) = \sqrt{\frac{1}{n}\sum_{h=1}^{L}\sum_{i\in U_h}(\rho_i - \bar{\rho}_h)^2} \ . \qquad (7.1)$$

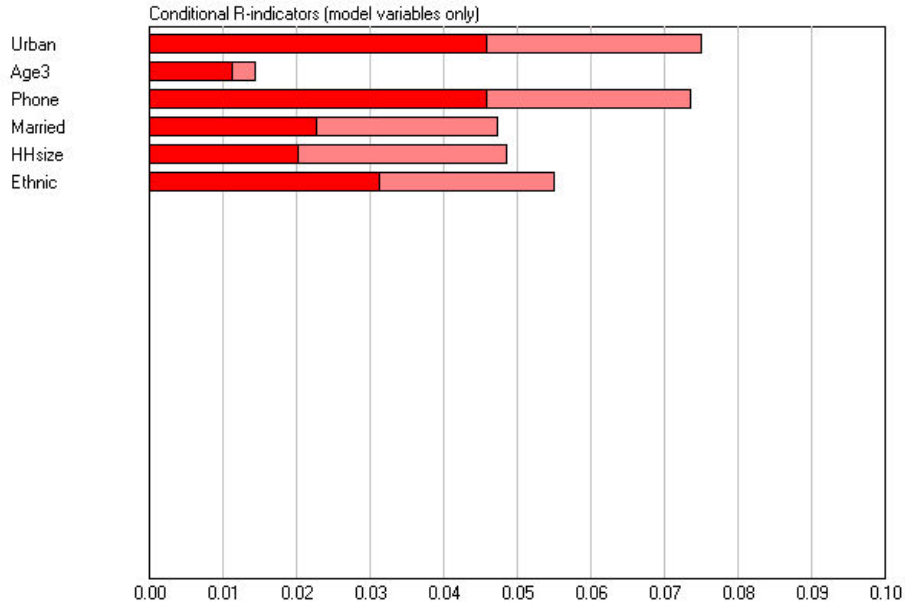
To say it in words: $P_C(X)$ is the remaining within cell variation of the response probabilities if the variable X is removed from the cross-classification. If, one the one hand, the remaining variation is small, the other variables are capable of explaining the variation. It can be concluded that there need not be a role for X in reducing the lack of representativity. If, on the other hand, the remaining variation is large, this can apparently not be accounted for by the other variables. So, there is an important role for X.

Also here it can be remarked that $P_C(X) \leq S(\rho) \leq 0.5$. i.e. the total variation within categories is smaller than the total variation.

Figure 7.1 shows the conditional partial indicators for the GPS data set. The variables shown are the variables included in the logit model for estimating the response probabilities. The plot shows both the unconditional (dark red) and conditional partial indicators (light red). The unconditional indicators are explained in section 5. The conditional indicators are always smaller than the unconditional indicators. It is clear that for the auxiliary variable *Urban* and *Phone* substantial within variation remains. Therefore, these are important variables for the treatment of the lack of respresentativity. *Age3* seems to be a less important variable.

A plot like this can be generated by the program by selecting *Conditional R-indicator* as the graph type and selecting no variable from the list of variables.

Figure 7.1. Conditional partial indicators

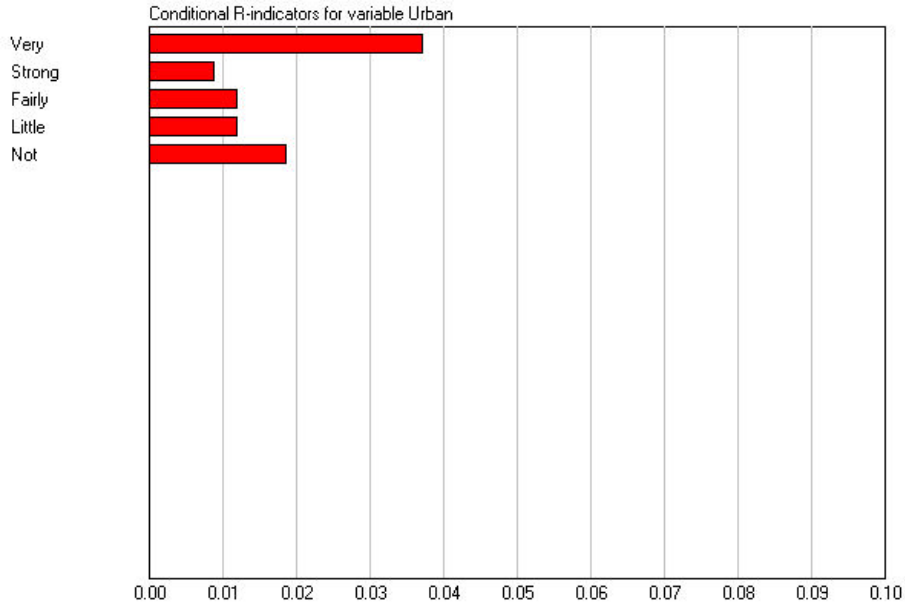


8. Conditional partial indicators at the category level

The conditional partial indicators as described in section 7 can give even more insight if they are computed for each category of a variable separately. The remaining within cell variation of the response probabilities after removing a variable *X* from the cross-classification, is computed for each category of *X* separately.

Figure 8.1 gives an example for the variable *Urban* in the GPS set data. It shows the conditional partial indicators for each category of *Urban*.

Figure 8.1. Conditional partial indicators within categories of a variable



The indicator has the largest value for the category *FstGenNonWest* (first generation non-western non-natives). For the treatment of the lack of representativity one should first look at this category.

Note that a large value does not necessarily means under-representation. It can also mean over-representation. So adjusting the fieldwork may mean do more or do less observations.

A plot like this can be generated by the program by selecting *Conditional R-indicator* as the graph type and selecting a variable from the list.

9. Input files

The program requires two files to be able to conduct an analysis:

- A *data file*. This file must contain the values of the variables for each sample element;
- A *metadata file*. This file must contain the names and value labels of the variables.

Figure 9.1 contains an example of a part of the sample data file *gps.dat*. The first 10 records of the GPS data are shown. Each line represents the record of a sample element. There are 14 variables: 13 auxiliary variables and a variable for the response probabilities. The response probability must always be the last value in the record.

Values of variables must be separated by one or more blanks. The values of the auxiliary variables must integers. The program can handle data sets with at most 20 variables. Each variable may have at most 20 categories. The maximum number of records that can be handled by the program, is 33,000.

Figure 9.1. Example of part of a data file

```
4 1 1 0 2 0 1 22 3 0 1 2 1 0.618489
1 1 1 0 4 0 1 25 4 0 0 3 1 0.527253
4 2 1 1 5 0 1 23 1 0 1 3 1 0.725618
4 3 1 1 4 0 1 24 1 0 1 3 1 0.685431
5 1 1 1 4 0 1 21 1 0 1 3 1 0.737672
4 2 1 0 4 0 2 23 5 0 1 4 1 0.636217
5 3 1 0 2 0 2 23 2 0 0 2 1 0.609066
5 2 1 0 1 0 1 21 1 0 1 1 1 0.618256
2 2 1 0 3 0 2 23 4 0 1 4 0 0.556704
1 3 0 0 1 0 1 21 2 0 0 1 1 0.328382
```

The metadata file describes the contents of the data file. Figure 9.2 contains part of the metadata file *gps.rin* for the GPS. Appendix A contains the complete contents of the file.

Figure 9.2. Example of part of a metadata file

```
Data
  gps.dat
Variables:13
  Urban:5,1
    1:Very
    2:Strong
    3:Fairly
    4:Little
    5:Not
  Age3:3,1
    1:18-34
    2:35-54
    3:55+
  Phone:2,1
    0:No
    1:Yes
```

The file must start with a line containing the keyword *Data*. Line 2 must contain the name of the data file. In this case, the name is *gps.dat*. Note that the indentation shown is not compulsory, but it enhances readability of the file,

Line 3 must contain the keyword *Variables*, followed by a colon, and the number of auxiliary variables. Note that the variable containing the response probabilities is excluded from this number. The GPS data set contains 13 auxiliary variables.

For each variable, there must be a line containing the name of the variable, followed by a colon, the number of categories, a comma, and a value 1 (if the variable is included in the model for the response probabilities) or 0 (the variable is not used in the model). So, the variable *Urban* has five categories and is included in the model.

The name of each variable is followed by a description of its categories. For each category, there is a line with the category code (as used in the data file), followed by a colon and a category label.

The program can handle data sets with at most 20 variables. Each variables may have at most 20 categories.

There is a tool for creating a data and metadata file from an SPSS file. This is an R program. It is described in appendix B and C. The program R must be installed in order to run the export tool.

There is a tool for creating a data and metadata file from a Stata file. This is an R program. It is described in appendix D and E. The program R must be installed in order to run the export tool.

Appendix A. Complete contents of the metadata file gps.rin.

```
Data
  gps.dat
Variables:13
  Urban:5,1
    1:Very
    2:Strong
    3:Fairly
    4:Little
    5:Not
  Age3:3,1
    1:18-34
    2:35-54
    3:55+
  Phone:2,1
    0:No
    1:Yes
  Married:2,1
    0:No
    1:Yes
  HHsize:5,1
    1:1
    2:2
    3:3
    4:4
    5:5+
  Ethnic:5,1
    0:Native
    1:FstGenNonWes
    2:FstGenWes
    3:SecGenNonWes
    4:SecGenWes
  Gender:2,0
    1:Male
    2:Female
  Region:5,0
    21:Greenfields
    22:Woodlands
    23:Lowlands
    24:Highlands
    25:Metropolis
  PNonNat1:8,0
    1:<5
    2:5-10
    3:10-15
    4:15-20
    5:20-30
    6:30-40
    7:40-50
    8:50+
  Allowan:2,0
    0:No
    1:Yes
  HasJob:2,0
    0:No
    1:Yes
  HHtype:5,0
    1:Single
    2:Couple
    3:CoupleChil
    4:SinglePar
    5:Other
  Response:2,0
    0:No
    1:Yes
```

Appendix B. Exporting data from SPSS to Cockpit

There is a tool available for exporting data and metadata from SPSS to files that can be read by Cockpit. This tool is written in R. It is called *export-spss.r*. Appendix C contains the code of this tool. This appendix describes how to adapt the code for a particular situation.

To apply the R program *export.r* in a specific situation some lines of the program have to be changed.

Step 1:
Specify the name of the SPSS file under the heading "Read SPSS file".:

```
spss.data = read.spss('d://cockpit/export/gps.por', use.value.label=FALSE)
```

Note that in the example in appendix A the SPSS file is not a standard SPSS system file (with extension sav), but a file in portable format (with extension por). This is because the R function read.spss sometimes has problems reading sav-files.

Step 2:
Specify the model variables, i.e. the variables used in the model for the response probabilities, under the heading "`Specify variables to export`":

```
model = c("URBAN", "AGE3", "PHONE", "MARRIED", "HHSIZE", "ETHNIC")
```

The names of these variables must be exactly equal to the SPSS variable names. The names are case-sensitive.

Step 3:
Specify the rest of the variables, i.e. the variables not used in the model for the response probabilities, under the heading "`Specify variables to export`":

```
rest  = c("GENDER", "REGION", "PNONNAT1", "ALLOWAN", "HASJOB", "HHTYPE", "RESPONSE")
```

The names of these variables must be exactly equal to the SPSS variable names. The names are case-sensitive. Note that not all other variables have to be specified, but just those that will be used in the graphical analysis.

Step 4:
Specify the variable containing the estimated response probabilities under the heading "`Specify variables to export`":

```
probs = "RESPROB"
```

The name of this variables must be exactly equal to the SPSS variable name. The name is case-sensitive.

Step 5:
Specify the names of the cockpit files to be created under the heading "`Specify variables to export`":

```
filename = "gps"
```

Two files will be created: a data file with the extension `dat`, and a metadata file with the extension `rin`. So, in this case there will be two files: `gps.dat` and `gps.rin`.

Appendix C. The program export-spss.r

```
#------------------------------------------------------------------------------
# Export SPSS file to Cockpit file
# export-spss version 1.0, 10 May 2010, by Jelke Bethlehem
# source("d://cockpit/test-export/export-spss.r")
#------------------------------------------------------------------------------

 rm(list=ls())
 library(foreign)

#------------------------------------------------------------------------------
# Read SPSS file
#------------------------------------------------------------------------------

 spss.data = read.spss('d://cockpit/test-export/gps.sav', use.value.label=FALSE)
 n = length(spss.data[[1]])

#------------------------------------------------------------------------------
# Specify variables to export
#------------------------------------------------------------------------------

 model = c("URBAN", "AGE3", "PHONE", "MARRIED", "HHSIZE", "ETHNIC")
 rest  = c("GENDER", "REGION", "PNONNAT1", "ALLOWAN", "HASJOB", "HHTYPE", "RESPONSE")
 probs = "RESPROB"
 filename = "d://cockpit/test-export/gps"

#------------------------------------------------------------------------------
# Read data
#------------------------------------------------------------------------------

 cat("Preparing data ...\n")
 f1 = paste(filename, ".dat", sep="")
 p1 = length(model)
 p2 = length(rest)
 p = p1 + p2
 p3 = p + 1
 data = matrix(rep(0, n * p3), n, p3)
 prob = rep(0, n)

 for (i in 1:p1)
 { col = spss.data[[model[i]]]
   data[,i] = as.integer(col)
 }
 for (i in 1:p2)
 { col = spss.data[[rest[i]]]
   data[,p1 + i] = as.integer(col)
 }
 prob = spss.data[[probs]]
 data[, p3] = as.real(prob)
 cat("Writing data ...\n")
 write(t(data), file=f1, ncolumns=p3)

#------------------------------------------------------------------------------
# Export metadata
#------------------------------------------------------------------------------

 cat("Writing metadata ...\n")
 f2 = paste(filename, ".rin", sep="")
 cat("Data", "\n", sep="", file=f2, append=FALSE)
 cat("  ", f1, "\n", sep="", file=f2, append=TRUE)
 cat("Variables:", p, "\n", sep="", file=f2, append=TRUE)
 for (i in 1:p1)
 { col = spss.data[[model[i]]]
   nam = model[i]
   lev = attr(col, "value.labels")
   lev = sort(lev)
   lab = names(lev)
   cod = as.integer(lev)
   cts = length(lev)
```

```
   cat("  ", nam, ":", cts, ",1" , "\n", sep="", file=f2, append=TRUE)
   for (j in 1:cts)
   { cat("    ", cod[j], ":", lab[j], "\n", sep="", file=f2, append=TRUE) }
 }
 for (i in 1:p2)
 { col = spss.data[[rest[i]]]
   nam = rest[i]
   lev = attr(col, "value.labels")
   lev = sort(lev)
   lab = names(lev)
   cod = as.integer(lev)
   cts = length(lev)
   cat("  ", nam, ":", cts, ",0" , "\n", sep="", file=f2, append=TRUE)
   for (j in 1:cts)
   { cat("    ", cod[j], ":", lab[j], "\n", sep="", file=f2, append=TRUE) }
 }

#-------------------------------------------------------------------------------
# End of program
#-------------------------------------------------------------------------------
```

Appendix D. Exporting data from Stata to Cockpit

There is a tool available for exporting data and metadata from Stata to files that can be read by Cockpit. This tool is written in R. It is called *export-stata.r*. Appendix E contains the code of this tool. This appendix describes how to adapt the code for a particular situation.

Note since SAS can store its data and metadata in a Stata file, this tool can also be used to export data and metadata from SAS to Cockpit.

To apply the R program *export-stata.r* in a specific situation some lines of the program have to be changed.

Step 1:
Specify the name of the Stata file under the heading "Read Stata file".:

```
stata.data = read.dta('d://cockpit/test-export/gps.dta')
```

Step 2:
Specify the model variables, i.e. the variables used in the model for the response probabilities, under the heading "`Specify variables to export`":

```
model = c("URBAN", "AGE3", "PHONE", "MARRIED", "HHSIZE", "ETHNIC")
```

The names of these variables must be exactly equal to the State variable names. The names are case-sensitive.

Step 3:
Specify the rest of the variables, i.e. the variables not used in the model for the response probabilities, under the heading "`Specify variables to export`":

```
rest  = c("GENDER", "REGION", "PNONNAT1", "ALLOWAN", "HASJOB", "HHTYPE", "RESPONSE")
```

The names of these variables must be exactly equal to the Stata variable names. The names are case-sensitive. Note that not all other variables have to be specified, but just those that will be used in the graphical analysis.

Step 4:
Specify the variable containing the estimated response probabilities under the heading "`Specify variables to export`":

```
probs = "RESPROB"
```

The name of this variables must be exactly equal to the Stata variable name. The name is case-sensitive.

Step 5:
Specify the names of the cockpit files to be created under the heading "`Specify variables to export`":

```
filename = "gps"
```

Two files will be created: a data file with the extension `dat`, and a metadata file with the extension `rin`. So, in this case there will be two files: `gps.dat` and `gps.rin`.

Appendix E. The program export-stata.r

```
#------------------------------------------------------------------------------
# Convert Stata file to Cockpit file
# convert-stata version 1.0, 10 May 2010, by Jelke Bethlehem
# source("d://cockpit/test-export/export-stata.r")
#------------------------------------------------------------------------------

 rm(list=ls())
 library(foreign)

#------------------------------------------------------------------------------
# Read Stata file
#------------------------------------------------------------------------------

 stata.data = read.dta('d://cockpit/test-export/gps.dta')
 n = length(stata.data[[1]])

#------------------------------------------------------------------------------
# Specify variables to export
#------------------------------------------------------------------------------

 model = c("URBAN", "AGE3", "PHONE", "MARRIED", "HHSIZE", "ETHNIC")
 rest  = c("GENDER", "REGION", "PNONNAT1", "ALLOWAN", "HASJOB", "HHTYPE", "RESPONSE")
 probs = "RESPROB"
 filename = "d://cockpit/test-export/gps"

#------------------------------------------------------------------------------
# Export data
#------------------------------------------------------------------------------

 cat("Preparing data ...\n")
 f1 = paste(filename, ".dat", sep="")
 p1 = length(model)
 p2 = length(rest)
 p = p1 + p2
 p3 = p + 1
 data = matrix(rep(0, n * p3), n, p3)
 prob = rep(0, n)

 for (i in 1:p1)
 { col = stata.data[[model[i]]]
   data[,i] = as.integer(col)
 }
 for (i in 1:p2)
 { col = stata.data[[rest[i]]]
   data[,p1 + i] = as.integer(col)
 }
 prob = stata.data[[probs]]
 data[, p3] = as.real(prob)
 ddd = t(data)

 cat("Writing data ...\n")
 write(ddd, file=f1, ncolumns=p3)

#------------------------------------------------------------------------------
# Export metadata
#------------------------------------------------------------------------------

 cat("Writing metadata ...\n")
 f2 = paste(filename, ".rin", sep="")
 cat("Data", "\n", sep="", file=f2, append=FALSE)
 cat("  ", f1, "\n", sep="", file=f2, append=TRUE)
 cat("Variables:", p, "\n", sep="", file=f2, append=TRUE)
 for (i in 1:p1)
 { col = stata.data[[model[i]]]
   nam = model[i]
   lev = levels(col)
   lab = lev
   cts = length(lev)
```

```
   cod = 1:cts
   cat("  ", nam, ":", cts, ",1" , "\n", sep="", file=f2, append=TRUE)
   for (j in 1:cts)
   { cat("    ", cod[j], ":", lab[j], "\n", sep="", file=f2, append=TRUE) }
 }
 for (i in 1:p2)
 { col = stata.data[[rest[i]]]
   nam = rest[i]
   lev = levels(col)
   lab = lev
   cts = length(lev)
   cod = 1:cts
   cat("  ", nam, ":", cts, ",0" , "\n", sep="", file=f2, append=TRUE)
   for (j in 1:cts)
   { cat("    ", cod[j], ":", lab[j], "\n", sep="", file=f2, append=TRUE) }
 }

#-------------------------------------------------------------------------------
# End of program
#-------------------------------------------------------------------------------
```