DP First Laboratory

Teresa Ricciardi   Albert Bertran

2023-03-10

3 Working with sdcMicro

3.1 Inspect a dataset and calculate k-anonymity using R

To start this laboratory the first step to do is to load the 'sdcMicro' package for statistical disclosure control methods, into our R session and then load the dataset 'free1' available in the sdcMicro package.

```
library(sdcMicro)
data("free1") # loads the dataset
```

The next step to do is to create a copy of the 'free1' dataset and coerce it to a data frame because many R functions and packages for data manipulation and analysis are designed to work specifically with data frames. Afterwards we check the class of 'newdataset' to be sure it is a data frame and we use the command to know the structure of it.

In addition we use the command 'attributes()' to obtain a list of attributes associated with the data frame and the command 'names()' to get a vector of the first four variable names in the data frame.

```
newdataset<-free1 # newdataset is a copy of the free1 dataset
newdataset<-as.data.frame(newdataset)
class(newdataset)
```

```
## [1] "data.frame"
```

```
str(newdataset) #new structure therefore, new attributes
```

```
## 'data.frame':    4000 obs. of  34 variables:
##  $ REGION  : num  36 36 36 36 36 36 36 36 36 36 ...
##  $ SEX     : num  1 1 1 1 2 2 2 2 1 2 ...
##  $ AGE     : num  43 27 46 27 24 24 34 26 26 37 ...
##  $ MARSTAT : num  4 4 4 4 4 4 4 4 1 1 ...
##  $ KINDPERS: num  3 3 1 1 1 1 1 1 2 2 ...
##  $ NUMYOUNG: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NUMOLD  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ AGEYOUNG: num  97 97 97 97 97 97 97 97 97 97 ...
##  $ EDUC1   : num  4 4 7 5 4 4 5 7 1 6 ...
##  $ EDUC2   : num  0 0 7 0 0 0 10 0 0 6 ...
##  $ ETNI    : num  1 9 1 1 1 1 1 1 9 1 ...
##  $ PRIOCCU : num  1 1 1 5 5 5 1 1 1 1 ...
##  $ POSLABM : num  1 1 1 1 4 4 1 1 1 1 ...
##  $ REGJOBC : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ RECBEN  : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ RECUNBEN: num  2 2 2 2 2 2 2 2 2 2 ...
##  $ RECODBEN: num  2 2 2 2 2 2 2 2 2 2 ...
##  $ RECBILL : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ RECSOSEC: num  2 2 2 2 2 2 2 2 2 2 ...
##  $ RECPENS : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ POSLABLY: num  7 7 7 7 2 2 7 7 7 7 ...
##  $ POSFACT : num  1 1 1 1 7 7 1 1 1 1 ...
##  $ COMPCODE: num  95 67 90 65 0 0 93 84 67 65 ...
##  $ OCCUCODE: num  300 552 219 481 0 0 593 90 531 411 ...
##  $ KINDFACT: num  8 8 8 8 97 97 8 8 8 8 ...
##  $ TENURE  : num  1 1 1 1 7 7 1 1 2 1 ...
##  $ FTPTIME : num  1 2 1 2 7 7 1 1 2 1 ...
##  $ ADDJOB  : num  2 2 2 2 7 7 2 2 2 2 ...
##  $ JOBFIND : num  2 2 2 2 7 7 2 1 1 2 ...
##  $ WEIGHT  : num  118 118 118 118 118 ...
##  $ INCOME  : num  970000 393700 152200 216100 566900 ...
##  $ MONEY   : num  74231 74231 74231 74231 74231 ...
##  $ ASSETS  : num  61340 90480 96600 40870 66460 ...
##  $ DEBTS   : num  1950 6750 6970 5950 4610 3900 5070 4240 4400 4620 ...
```

```r
attributes(newdataset)
```

```
## $names
##  [1] "REGION"   "SEX"      "AGE"      "MARSTAT"  "KINDPERS" "NUMYOUNG"
##  [7] "NUMOLD"   "AGEYOUNG" "EDUC1"    "EDUC2"    "ETNI"     "PRIOCCU"
## [13] "POSLABM"  "REGJOBC"  "RECBEN"   "RECUNBEN" "RECODBEN" "RECBILL"
## [19] "RECSOSEC" "RECPENS"  "POSLABLY" "POSFACT"  "COMPCODE" "OCCUCODE"
## [25] "KINDFACT" "TENURE"   "FTPTIME"  "ADDJOB"   "JOBFIND"  "WEIGHT"
## [31] "INCOME"   "MONEY"    "ASSETS"   "DEBTS"
##
## $class
## [1] "data.frame"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
##  [ reached getOption("max.print") -- omitted 3900 entries ]
```

a) Checking the output of the following command we can see the first 4 variables of the dataset being: Region, Sex, Age, Marstat. Therefore, taking into account their meaning, Region, Sex and Marital Status are Categorical variables while Age would be a continuous variable.

```r
names(newdataset)[1:4]
```

```
## [1] "REGION"  "SEX"     "AGE"     "MARSTAT"
```

b) At this point we have to convert the four variables above to a factor because we want to represent them as categorical variables and then we create vectors that contain the levels of the four variables.

```
newdataset$REGION<-as.factor(newdataset$REGION)
newdataset$SEX<-as.factor(newdataset$SEX)
newdataset$AGE<-as.factor(newdataset$AGE)
newdataset$MARSTAT<-as.factor(newdataset$MARSTAT)
# Levels of the region parameter
r = levels(newdataset$REGION)
# Levels of the sex parameter
s = levels(newdataset$SEX)
# Levels of the age parameter
a = levels(newdataset$AGE)
# Levels of the marital status parameter
m = levels(newdataset$MARSTAT)

length(r)*length(s)*length(a)*length(m)
```

```
## [1] 87360
```

After having done the last operation, we obtain that the number of possible keys that can be performed with these four variables is '87360', and comparing this value with the number of records in the dataset (4000) we can say that, since the number of possible keys is much higher that the number of records it is less likely that there will be rare keys that are easy to disclosure.

Now we have to create a contingency table with the REGION and SEX variables, and then another with the REGION, SEX, AGE and MARSTAT variables, and we use the command 'sum()' to see how many keys violate the 2-anonymity and the 3-anonymity condition. The results are shown below with the use of the function 'freqCalc()'.

```
contable = table(newdataset$REGION,newdataset$SEX)

sum(contable[contable < 2])
```

```
## [1] 15
```

```
sum(contable[contable < 3])
```

```
## [1] 49
```

```
contable4 = table(newdataset$REGION,newdataset$SEX,newdataset$AGE,newdataset$MARSTAT)

sum(contable4[contable4 < 2])
```

```
## [1] 3014
```

```
sum(contable4[contable4 < 3])
```

```
## [1] 3738
```

```r
freqCalc(newdataset,keyVars = c("REGION","SEX"))
```

```
##
##  --------------------------

## 15 obs. violate 2-anonymity

## 49 obs. violate 3-anonymity

##  --------------------------
```

```r
freqCalc(newdataset,keyVars = c("REGION","SEX","AGE","MARSTAT"))
```

```
##
##  --------------------------

## 3014 obs. violate 2-anonymity

## 3738 obs. violate 3-anonymity

##  --------------------------
```

The aim of the following part of the laboratory is to inspect the dataset and to do that we have to create an sdcObject. As a first step we have to create a new data frame from the data set 'free1' and then create the object 'sdc' and convert the variables to factors. After that we use the command 'print()' with the correct type options to obtain the re-identification risk and the k-anonymity information that are shown below.

```r
dataset32<-free1

dataset32<-as.data.frame(dataset32)

sdc <- createSdcObj(
  dat = dataset32,
  keyVars = c("REGION","SEX","AGE","MARSTAT")
)
sdc <- varToFactor(sdc, "REGION")
sdc <- varToFactor(sdc, "SEX")
sdc <- varToFactor(sdc, "AGE")
sdc <- varToFactor(sdc, "MARSTAT")

print(sdc, type="risk")
```

```
## Risk measures:
##
## Number of observations with higher risk than the main part of the data: 0
## Expected number of re-identifications: 3450.00 (86.25 %)
```

```
print(sdc, type="kAnon")
```

```
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##   - 2-anonymity: 3014 (75.350%)
##   - 3-anonymity: 3738 (93.450%)
##   - 5-anonymity: 3943 (98.575%)
##
## ----------------------------------------------------------------------
```

Then we check the available slots of the object and we access the slot 'risk'.

```
slotNames(sdc)
```

```
##  [1] "origData"           "keyVars"            "pramVars"
##  [4] "numVars"            "ghostVars"          "weightVar"
##  [7] "hhId"               "strataVar"          "sensibleVar"
## [10] "manipKeyVars"       "manipPramVars"      "manipNumVars"
## [13] "manipGhostVars"     "manipStrataVar"     "originalRisk"
## [16] "risk"               "utility"            "pram"
## [19] "localSuppression"   "options"            "additionalResults"
## [22] "set"                "prev"               "deletedVars"
```

```
sdc@risk
```

```
## $global
## $global$risk
## [1] 0.8625
##
## $global$risk_ER
## [1] 3450
##
## $global$risk_pct
## [1] 86.25
##
## $global$threshold
## [1] 0
##
## $global$max_risk
## [1] 0.01
##
##
## $individual
##              risk fk Fk
##    [1,] 0.5000000  2  2
##    [2,] 0.5000000  2  2
##    [3,] 1.0000000  1  1
##    [4,] 0.5000000  2  2
##    [5,] 0.1666667  6  6
##    [6,] 0.1666667  6  6
##    [7,] 0.2500000  4  4
##     [8,] 0.1428571  7  7
##     [9,] 1.0000000  1  1
##    [10,] 1.0000000  1  1
##    [11,] 1.0000000  1  1
##    [12,] 0.5000000  2  2
##    [13,] 1.0000000  1  1
##    [14,] 0.1428571  7  7
##    [15,] 0.1428571  7  7
##    [16,] 0.2500000  4  4
##    [17,] 0.2500000  4  4
##    [18,] 0.5000000  2  2
##    [19,] 0.2000000  5  5
##    [20,] 0.2000000  5  5
##    [21,] 0.2000000  5  5
##    [22,] 1.0000000  1  1
##    [23,] 0.1666667  6  6
##    [24,] 1.0000000  1  1
##    [25,] 0.3333333  3  3
##    [26,] 0.5000000  2  2
##    [27,] 0.5000000  2  2
##    [28,] 0.5000000  2  2
##    [29,] 0.3333333  3  3
##    [30,] 0.1666667  6  6
##    [31,] 1.0000000  1  1
##    [32,] 0.3333333  3  3
##    [33,] 0.5000000  2  2
##  [ reached getOption("max.print") -- omitted 3967 rows ]
```

4 SDC methods:

4.1 Recoding:

a) First of all, we create a list with all the regions, and a list with the different values of the regions list defined. This two lists will be used later on to recode the regions values.

```
regions <- sdc@manipKeyVars$REGION
diff_regions <- levels(regions)
```

Now, we construct a list with the different categories: North(45), South(45), Est(45) and West(total-135) following the instructions in the statement.

```
groups <- c(rep("NORTH",45), rep("SOUTH",45), rep("EST",45), rep("WEST", length(diff_regions)-135))
```

Then we just need to use the groupAndRename function specifying the previous values, being the numbers from 1 to 182, and the new ones we want to convert to being the cardinal directions defined above. The groupAndRename function will swap each element from the before list for the value in the after list in the region column in the sdc object.

```
nRegions <- groupAndRename(obj = sdc, var = c("REGION"), before = c(diff_regions), after = c(groups))
```

Now we can construct a table to count how many different cardinal points we have.

```
table(nRegions@manipKeyVars$REGION)
```

```
##
## NORTH SOUTH   EST  WEST
##  1143  1020  1040   797
```

Now, just to check that everything worked properly we do the following checks:

```
checkCorrectness <- table(sdc@manipKeyVars$REGION)
sum(checkCorrectness[1:45])
```

```
## [1] 1143
```

```
sum(checkCorrectness[46:90])
```

```
## [1] 1020
```

```
sum(checkCorrectness[91:135])
```

```
## [1] 1040
```

```
sum(checkCorrectness[136:length(diff_regions)])
```

```
## [1] 797
```

b) We can now check the values of 2/3 anonymity and see how they have been modified when applying our changes.

```
print(nRegions, type="kAnon")
```

```
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##   - 2-anonymity: 282 (7.050%) | in original data: 3014 (75.350%)
##   - 3-anonymity: 538 (13.450%) | in original data: 3738 (93.450%)
##   - 5-anonymity: 1090 (27.250%) | in original data: 3943 (98.575%)
##
## ----------------------------------------------------------------------
```

c) Since in the previous exercises we swap the variable type of "AGE" to factor we now need to convert it back to numeric.

```
sdc <- varToNumeric(obj = sdc, var = "AGE")
```

Since we now want to change the variable "AGE" for a category, like: Children, Young, Adults and Senior, here we define the interval of each category being the followings:

- (0-14)
- (15-25)
- (25-64)
- (65-Inf)

```
intervals <- c(0,14,25,64,Inf)
```

Now we can to the recode in a similar way as we did before, specifying the intervals above and the values that will get swapped, being:

- Children.
- Young.
- Adults.
- Senior.

```
nAge <- globalRecode(obj = sdc, column = c("AGE"), intervals, labels = c("Children","Young","Adults","S
```

After recoding we can check how many persons do we have for each interval of age.

```
table(nAge@manipKeyVars$AGE)
```

```
##
## Children    Young   Adults   Senior
##        0      767     2754      479
```

As we did before when swapping the region to cardinal points, we can check the correctness of the previous steps with the following command, checking the different ages in the original data.

```
table(sdc@manipKeyVars$AGE)
```

```
##
##  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
##  56  73  61  61  61  73  69  73  82  80  78  72  87  95 106 109  85  93  85 100
##  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
## 108 103  88 106  91  85  82  72  97  93  54  66  57  52  59  41  44  43  46  52
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74
##  37  41  58  47  46  50  52  53  51  48  50  70  38  52  47  62  35  36  52  37
```

d) We can now check the new values of k-anonymity after applying the previous transformation.

```
print(nAge, type="kAnon")
```

```
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##   - 2-anonymity: 566 (14.150%) | in original data: 3014 (75.350%)
##   - 3-anonymity: 1012 (25.300%) | in original data: 3738 (93.450%)
##   - 5-anonymity: 1805 (45.125%) | in original data: 3943 (98.575%)
##
## ----------------------------------------------------------------------
```

4.2 Local Suppression:

a) Before starting the GUI we just check if the dataset is a data.frame and it actually is, so we do not need to modify it.

```
class(newdataset)
```

```
## [1] "data.frame"
```

Now we can open the GUI and start doing the exercise.

```
#sdcApp()
```

b) The following is the code that creates the sdcObject in the GUI with the categorical variables specified.

```
library(sdcMicro)

obj <- NULL
if (!exists("newdataset")) {
  stop('object "newdataset" is missing; make sure it exists.`', call. = FALSE)
}
obj$inputdata <- readMicrodata(path="newdataset", type="rdf", convertCharToFac=FALSE, drop_all_missings=
inputdataB <- obj$inputdata

## Set up sdcMicro object
sdcObj <- createSdcObj(dat=inputdata,
    keyVars=c("REGION","SEX","AGE","MARSTAT"),
    numVars=NULL,
    weightVar=NULL,
    hhId=NULL,
    strataVar=NULL,
    pramVars=NULL,
    excludeVars=NULL,
    seed=0,
    randomizeRecords=FALSE,
    alpha=c(1))

## Store name of uploaded file
opts <- get.sdcMicroObj(sdcObj, type="options")
opts$filename <- "newdataset"
sdcObj <- set.sdcMicroObj(sdcObj, type="options", input=list(opts))
```

c) The following is the code for computing and checking the k-anonymity with k values being 3 or 5 and with the following order of categorical variables when computing the k-anonymity: SEX -> MARSTAT -> AGE -> REGION.

```
## Local suppression to obtain k-anonymity
sdcObj <- kAnon(sdcObj, importance=c(4,1,3,2), combs=NULL, k=c(3))
sdcObj <- undolast(sdcObj)
## Local suppression to obtain k-anonymity
sdcObj <- kAnon(sdcObj, importance=c(4,1,3,2), combs=NULL, k=c(5))
sdcObj <- undolast(sdcObj)
```

The following is a table with the suppressed values when applying k-anonymity and the time it took to compute it:

Suppressed values	3-Anonymity	5-Anonymity
Region	3701	3870
Sex	0	0
Age	38	105
Marstat	0	0
Total time	7.91s	16.14s
Time (prev-new)	7.91s	8.23s

Note that the most suppressed values are Region and Age, the ones with most different values and therefore the ones that can identify the most. Since Marstat has only 4 possible values and Sex 2. Also note that as the value of k increases, the computation time to achieve k-anonymity also increases.

d) The following is the same code as above but changing the importance of the categorical variables when computing the k-anonymity, now being: REGION -> AGE -> MARSTAT -> SEX.

```
## Local suppression to obtain k-anonymity
sdcObj <- kAnon(sdcObj, importance=c(1,4,2,3), combs=NULL, k=c(3))
sdcObj <- undolast(sdcObj)
## Local suppression to obtain k-anonymity
sdcObj <- kAnon(sdcObj, importance=c(1,4,2,3), combs=NULL, k=c(5))
```

The following is a table with the suppressed values when applying k-anonymity and the time it took to compute it with the following order of variables: Region -> Age -> Marstat -> Sex.

Suppressed values	3-Anonymity	5-Anonymity
Region	10	10
Sex	1849	2174
Age	1220	1351
Marstat	855	1213
Total time	34.18s	57.09s
Time (prev-new)	18.04s	22.91s

In this second table, we can see that instead of just suppressing 2 variables as we did before, we now suppress data from all the 4 variables. This is due to the fact, that in the previous section we start by grouping Sex and Marstat and it's difficult to identify someone only by these variables, therefore no suppression is needed till we group with age and some identifications can be done. But in this section, as we group the Region with the Age we get already some identifications, this is why we need to suppress values of each variable.

The main conclusion is that when we apply local suppression, we should consider starting by the variables that have less possible values and therefore less restrictives since we would need to suppress less values and the computation time will be faster.

e) Finally, in this last exercise we link the variable RECBEN with the REGION variable and compute the 10-anonymity with the following priority: AGE -> MARSTAT -> SEX -> RECBEN.

```r
## Adding linked (ghost)-Variables
sdcObj <- addGhostVars(sdcObj, keyVar="RECBEN", ghostVars=c("REGION"))
## Local suppression to obtain k-anonymity
sdcObj <- kAnon(sdcObj, importance=c(3,1,2,4), combs=NULL, k=c(10))
```

The next table represents the number of suppressions in the variables RECBEN and Region and as can be seen, apart from reducing the number of suppressed values in the previous sections, since the Region got related with RECBEN, we can see that the values are the same since they are linked together.

Suppressed values	Before	After
RECBEN	0	437
REGION	0	437