DP First Laboratory

Teresa Ricciardi Albert Bertran

2023-03-07

```
library(sdcMicro)
```

```
# we can use the instruction data() to show the available datasets
data("free1") # loads the dataset
#we use <- to assign a value to a variable
newdataset<-free1 # newdataset is a copy of the free1 dataset
```

```
newdataset<-as.data.frame(newdataset)
class(newdataset)
str(newdataset) #new structure therefore, new attributes
attributes(newdataset)
# ?data.frame  # you can use a question mark before a command to obtain some help
```

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

Checking the output of the previous commands we can see the first 4 variables of the dataset being: Region, Sex, Age, Marstat. So that, Region, Sex and Marital Status are Categorical while Age would be continuous.

```
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)
```

Easy to disclure.

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

```
print(contable)
```

```
sum(contable[contable < 2])
sum(contable[contable < 3])
```

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

```
sum(contable4[contable4 < 2])
sum(contable4[contable4 < 3])
```

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

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

## 15 obs. violate 2-anonymity

## 49 obs. violate 3-anonymity

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

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

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

## 3014 obs. violate 2-anonymity

## 3738 obs. violate 3-anonymity

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

```
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%)
##
## ----------------------------------------------------------------------
```

```
slotNames(sdc)
```

```
sdc@risk
```

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)
```

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

```
checkCorrectness <- table(sdc@manipKeyVars$REGION)
sum(checkCorrectness[1:45])
sum(checkCorrectness[46:90])
sum(checkCorrectness[91:135])
sum(checkCorrectness[136:length(diff_regions)])
```

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)
```

??

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)
```

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

```
#sdcApp()
```

```
#b)

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))
```

b)

```
## 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)
```

c)

```
## 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))
```

d)

```
## 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))
```

e)