

DP Second Laboratory

Teresa Ricciardi

Albert Bertran

2023-04-10

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

1. PRAM (Post Randomization Method)

a) Use `table()` to obtain the frequencies for each category of the MARSTAT variable of the `free1` dataset. Define your own P matrix (remember the sum of the rows must be 1) and calculate the theoretical expected frequency for each category after “praming” the MARSTAT variable. Are the expected values of the frequencies after “praming” similar to the original ones? First, we create a newdataset from the “free1” dataset and we check the different values MARSTAT has.

```
newdataset<-free1
v <- table(newdataset[, "MARSTAT"])
v
```

```
##
##      1      2      3      4
## 2547  162  171 1120
```

We now create a P matrix to use for praming. The values used are just the ones in the slides of the session.

```
mdat <- matrix(c(0.5,0.4,0.1,0,
                 0.1,0.6,0.2,0.1,
                 0.1,0.3,0.6,0,
                 0,0.1,0.2,0.7), nrow = 4, byrow = TRUE,
              dimnames = list(c("single", "married", "divorced", "widow"),
                             c("single", "married", "divorced", "widow")))
mdat
```

```
##           single married divorced widow
## single      0.5      0.4      0.1    0.0
## married     0.1      0.6      0.2    0.1
## divorced    0.1      0.3      0.6    0.0
## widow       0.0      0.1      0.2    0.7
```

Now, using `crossprod` we can compute the new values, noting that they differ a lot from the original ones, so the matrix used is not ideal.

```
crossprod(v, mdat)
```

```
##      single married divorced widow
## [1,] 1306.8  1279.3    613.7 800.2
```

We now convert the dataset to a dataframe to access it in the `sdApp`.

```
newdataset<-as.data.frame(newdataset)
```

b) Check your result using `sdApp()`. Setup a DSC problem using the `free1` dataset selecting `REGION`, `SEX` and `AGE` as categorical variables and `MARSTAT` as the variable to be “pramed”. To obtain always the same result with probabilistic methods use the same seed. (Don’t worry about parameter `alpha`). Obtain the frequencies for each category of the variable `MARSTAT` (“Microdata/Explore variables” option). Use `PRAM` (expert) option in the “Anonymize” menu to “pram” `MARSTAT` with the matrix that you have defined in the previous section. Check the new frequencies (“Anonymize/Explore variables” option).

```
#sdApp()
```

Age/Marstat	1	2	3	4
15	1	0	0	55
16	1	0	0	72
17	2	0	0	59
18	0	0	0	61

c) Undo the last step and go to the “Anonymize/PRAM (simple)” option to create an invariant probability transition matrix and “pram” the variable `MARSTAT` (use the default values `pd=0.8` and `alpha=0.5`). Create a table comparing the frequencies and percentages of the `MARSTAT` variable before and after “praming” (use “Explore variables” in “Microdata” and “Anonymize” menus). Are the frequencies similar?

d) Create an `sdObject` like the one in the previous sections and “pram” the `MARSTAT` variable. Get the `PRAM` array from the “pram” slot of the `sdObject` (e.g. `P<-sdObj@pramparamsMARSTAT$Rs`) and perform the following calculation:

- Transpose the matrix and calculate the eigenvalues and the eigenvectors
- Check that 1 is one of the eigenvalues
- Normalize its associated eigenvector so that the sum of its components is 1.
- Compare the eigenvector with the percentages of the original values of `MARSTAT`. Drawn your own conclusions.

2. Microaggregation

a) Compare the univariate, multivariate simple, and `mdav` microaggregation algorithms.

```

dataset32<-free1

dataset32<-as.data.frame(dataset32)

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

```

1. Create an sdc object with the free1 dataset using REGION, SEX, AGE and MARSTAT as categorical variables and INCOME, ASSETS and DEBTS as numeric variables.

```

# mdav, single, onedims
print("Mdav algorithm time:")

```

2. Execute the different microaggregation algorithms using $K = 4$ and compute the execution time for each of them. (K is the minimum size of each group).

```
## [1] "Mdav algorithm time:"
```

```

startMdav <- Sys.time()
mdavSdc <- microaggregation(obj = sdc, method = "mdav", aggr = 4)
endMdav <- Sys.time()
mdav_time <- round(endMdav-startMdav,4)
mdav_time

```

```
## Time difference of 0.1068 secs
```

```
print("Single algorithm time:")
```

```
## [1] "Single algorithm time:"
```

```

startSingle <- Sys.time()
singleSdc <- microaggregation(obj = sdc, method = "single", aggr = 4)
endSingle <- Sys.time()
single_time <- round(endSingle-startSingle,4)
single_time

```

```
## Time difference of 0.3207 secs
```

```
print("Onedims algorithm time:")
```

```
## [1] "Onedims algorithm time:"
```

```
startOnedims <- Sys.time()
onedimsSdc <- microaggregation(obj = sdc, method = "onedims", aggr = 4)
endOnedims <- Sys.time()
onedims_time <- round(endOnedims-startOnedims,4)
onedims_time
```

```
## Time difference of 0.0705 secs
```

```
mdav_risk <- mdavSdc@risk$numeric
single_risk <- singleSdc@risk$numeric
onedims_risk <- onedimsSdc@risk$numeric

mdav_util <- mdavSdc@utility$il1
single_util <- singleSdc@utility$il1
onedims_util <- onedimsSdc@utility$il1

mdav_eigen <- mdavSdc@utility$eigen
single_eigen <- singleSdc@utility$eigen
onedims_eigen <- onedimsSdc@utility$eigen
```

```
# We can now create the dataframe with the results obtained.
```

```
algorithm <- c("MDAV", "Single", "Onedims")
time <- c(mdav_time, single_time, onedims_time)
risk <- c(mdav_risk, single_risk, onedims_risk)
utility <- c(mdav_util, single_util, onedims_util)
eigen <- c(mdav_eigen, single_eigen, onedims_eigen)

df_results <- data.frame(algorithm, time, risk, utility, eigen)

print(df_results)
```

3. From the new__sdc object get the following risk and utility measurements.

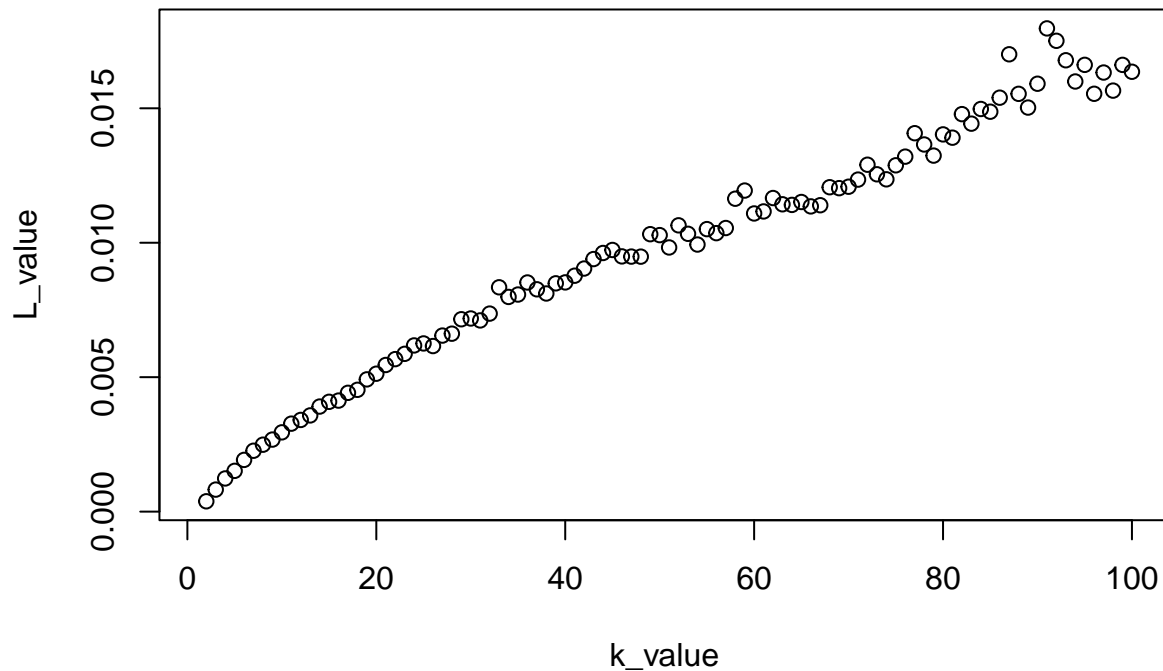
```
##   algorithm      time      risk      utility      eigen
## 1    MDAV 0.1068 secs 0.05875  54903.6037 0.0001568712
## 2   Single 0.3207 secs 0.00000  431155.9111 0.0269553104
## 3  Onedims 0.0705 secs 1.00000    770.4681 0.0001238770
```

From the results presented above, we can justify why MDAV is the most widely used microaggregation algorithm since it achieves a good balance between risk and utility. It has a lower risk value, indicating that it preserves more of the original data, and a lower utility, indicating that it distorts the data less. Regarding the eigen column measuring the loss of correlation between the variables and MDAV also performs well. Additionally, MDAV also has a reasonable running time, resulting a good choice when choosing a microaggregation algorithm.

b) In order to check the trade-off between risk and utility, use the `sdc` object created in the previous section and run `mdav` for values of `K` from 2 to 100. Calculate the `L` parameter in each case. Plot `K` vs `L`.

```
histogram = c()

original <- sdc@origData[,sdc@numVars]
meanOrig <- c(mean(original[,1]), mean(original[,2]), mean(original[,3]))
sst <- sum((original - meanOrig)^2)
for (k in 2:100) {
  sdcMdavFor <- microaggregation(obj = sdc, method = "mdav", aggr = k)
  manip <- sdcMdavFor@manipNumVars
  sse <- sum((original - manip)^2)
  L <- sse / sst
  histogram[k] <- L
}
plot(histogram, xlab = "k_value", ylab = "L_value")
```



Finally we can observe the correlation between the `K` and `L` value obtained when increasing the value of `K`.