Advanced Survey Statistics: Disclosure Control

Part 5: Anonymisation Methods

Matthias Templ

Institut für Datenanalyse und Prozessdesign
School of Engineering
Zürcher Hochschule für Angewandte Wissenschaften

FU-Berlin, 2019

Traditional anonymisation methods

We will discuss these standard methods

Methods that recode data or suppress values

- global recoding
- local suppression
- microaggregation

Methods that include a probability mechanism

- PRAM
- adding noise

Recoding

Recoding categorical key variables:

- achieving anonymity by mapping the values of the categorical key variables to generalized or altered categories.
- Example: combine multiple levels of schooling (e.g., secondary, tertiary, postgraduate) into one level (e.g., secondary and above).

Recoding continuous variables

- means to discretize the variable
- Example: income to income classes

Recoding

Test data from the Philippines (household income data):

```
require("sdcMicro")
data(testdata, package="sdcMicro")
testdata$urbrur <- factor(testdata$urbrur)
testdata$water <- factor(testdata$water)
testdata$relat <- factor(testdata$relat)
testdata$walls <- factor(testdata$walls)
sdc <- createSdcObj(testdata,
          keyVars=c('urbrur','water','sex','age','relat'),
          numVars=c('expend','income','savings'),
          pramVars=c("walls"),
          w='sampling_weight',
          hhId='ori_hid', alpha = 0.7)
summary(testdata$age)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.00    9.00   19.00   24.11   36.00   95.00
```

Recoding

```
labs <- c("1-9","10-19","20-29","30-39",
          "40-49","50-59","60-69","70-79","80-130")
sdc <- globalRecode(sdc, column="age",
                    breaks=c(0,9,19,29,39,49,59,69,79,130),
                    labels=labs)
print(sdc)
```

```
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##   - 2-anonymity: 113 (2.467%) | in original data: 653 (
##   - 3-anonymity: 188 (4.105%) | in original data: 1087
##   - 5-anonymity: 362 (7.904%) | in original data: 1781
##
## ---------------------------------------------------------
```

To combine specific categories use `groupAndRename`.

Recoding

```
sdc <- groupAndRename(sdc, var="water",
         before=levels(testdata$water),
         after=c("1","2","3","4","5","6-9","6-9","6-9"))
sdc <- groupAndRename(sdc, var="relat",
         before=levels(testdata$relat),
         after=c("1","2","3","4","5","6","7","8-9","8-9"))
print(sdc, "kAnon")
```

```
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##   - 2-anonymity: 106 (2.314%) | in original data: 653 (1
##   - 3-anonymity: 171 (3.734%) | in original data: 1087 (
##   - 5-anonymity: 316 (6.900%) | in original data: 1781 (
##
## ----------------------------------------------------------
```

Top- and Bottom Coding

Top (Bottom) Coding

- continuous variables are cut off by a given upper (lower) threshold
- values above (below) are replaced with, e.g., the mean of values above (below) the threshold

```
sdc <- topBotCoding(sdc, value=500000, replacement=500000,
                          column="income")
```

- advantage: easy to explain and thus often applied
- disadvantage: the highest (lowest) values are then identical
- extension to the multivariate case: see presentation on Wednesday

Local suppression

- Typically used after recoding to minimize residual risk.
- Heuristic optimization methods to find specific patterns in categorical key variables. Replace this pattern with missing values.
- One aim: to **suppress a minimum amount of values** and in the same time **guarantee k-anonymity**.
- Additional complexity: frequency counts with missing Values.
- Weight the variables according to their importance (in some variables you may want to end with less suppresions than in some others)

Local suppression - approaches

Mondrian: combine categories to achieve k-anonymity by a recoding strategy based on counts of categories. Too over-simplistic approach, not very promising results because the algorithm combines categories without asking their meaning.

all-M approach: whenever k-anonymity cannot be provided because of having too many key variables, then k-anonymity is provided in all subsets of size M of the key variables. More precisely, the algorithm will provide k-anonymity for each combination of M key variables.

k-anonymity approach: ensures k-anonymity for the combination of all key variables. If the number of key variables is too high: all-M approach or PRAM (next method to be explained) for specific key variables.

Local suppression

```
# all M approach
combs <- 5:3
k <- c(10,20,30)
sdc <- kAnon(sdc, k = k, combs = combs, importance = c(3,4,
# print(sdc, "kAnon")
print(sdc, "ls")
```

```
## Local suppression:
##  KeyVar | Suppressions (#) | Suppressions (%)
##  urbrur |               82 |            1.790
##   water |              533 |           11.638
##     sex |                6 |            0.131
##     age |                0 |            0.000
##   relat |              545 |           11.900
## ---------------------------------------------------------
```

```
sdc <- undolast(sdc)
```

Local suppression

```r
# k-anonymity for all key variables
sdc <- kAnon(sdc, k = 3, importance = c(3,4,2,1,5))
# print(sdc, "kAnon")
print(sdc, "ls")
```

```
## Local suppression:
##  KeyVar | Suppressions (#) | Suppressions (%)
##  urbrur |                2 |            0.044
##   water |               20 |            0.437
##     sex |                0 |            0.000
##     age |                0 |            0.000
##   relat |              157 |            3.428
## ----------------------------------------------------------------------
```

kAnon in subsets

Stratification

- The methods can also be applied on each strata separately as long as the strata is specified in `createSdcObj`.
- Automatically then the algorithm ensures *k*-anonymity in all strata.

PRAM

Especially if the number of categorical key variables is large or many of these variables have a high number of different categories - recoding and local suppression would modify the data too much.

- PRAM is applied to one variable at a time.
- We swap values between categories with pre-defined probabilities.
- An attacker can never be sure if a value is true or has been swapped.
- Probabilities for swapping values are chosen to be small in practice.
- In practice: often the geographical information is swapped using PRAM

PRAM Example

Consider a variable *location* with categories east, middle, west.

- We define a 3-by-3 transition matrix with p_{ij} the probabilities for swapping category i to j. $\sum_{j=1}^{3} p_{ij} = 1 \quad ,\forall i \in \{1,2,3\}$.
- For example, the matrix could look like this:

$$\mathbf{P} = \begin{pmatrix} 0.9 & 0.05 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \end{pmatrix}$$

- $\rightarrow$ the probability that a value stays the same is 0.9, because $p_{11} = p_{22} = p_{33}$
- The probability that east will become middle is $p_{12} = 0.05$
- ...

PRAM Example

```
sdc <- pram(sdc) # with standard defaults
print(sdc, "pram")
```

```
## Post-Randomization (PRAM):
## Variable: walls
## --> final Transition-Matrix:
##             2         3           9
## 2 0.976466014 0.0149648 0.008569182
## 3 0.005411079 0.9944792 0.000109713
## 9 0.206174522 0.0073003 0.786525178
##
## Changed observations:
##   variable nrChanges percChanges
## 1    walls        59        1.29
## ----------------------------------------------------
```

How to build an own transition matrix: ?pram

Microaggregation

For continuous key variables. Clustering of observations into groups and replace values with group means.

- Observations should be as similar as possible within a group.
- Problem is NP-hard. Heuristic algorithm: MDAV
- Assign an aggregated value in each group.
  - arithmetic mean or e.g. robust means
- Application typically applied independently in subgroups, eg, independent in all regions.
- also version for mixed-scaled variables available (Gower)

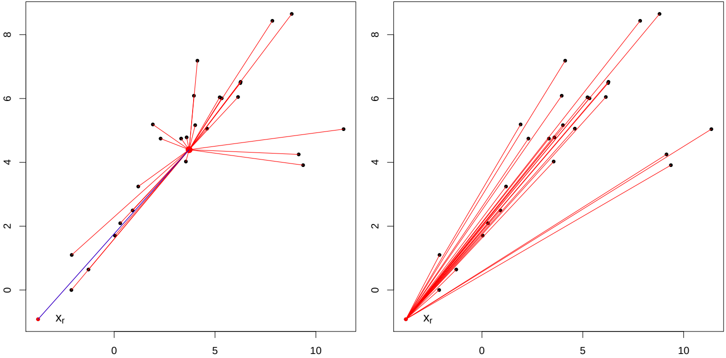
Microaggregation Example

Example with aggregation level 2

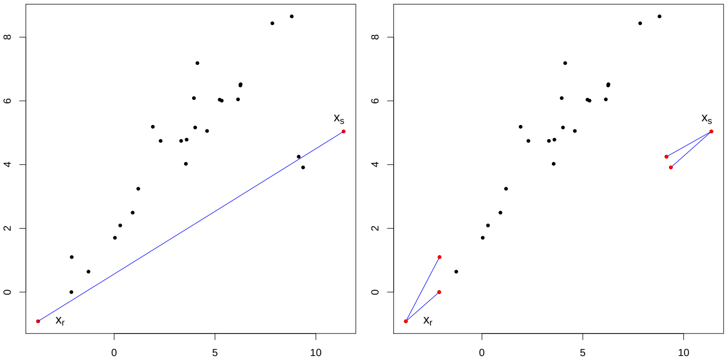
Tabelle 1: Example of micro-aggregation. Columns 1-3 contain the original variables, columns 4-6 the micro-aggregated values (rounded on two digits).

	Num1	Num2	Num3	Mic1	Mic2	Mic3
1	0.30	0.400	4	0.65	0.85	8.5
2	0.12	0.220	22	0.15	0.51	15.0
3	0.18	0.800	8	0.15	0.51	15.0
4	1.90	9.000	91	1.45	5.20	52.5
5	1.00	1.300	13	0.65	0.85	8.5
6	1.00	1.400	14	1.45	5.20	52.5
7	0.10	0.010	1	0.12	0.26	3.0
8	0.15	0.500	5	0.12	0.26	3.0

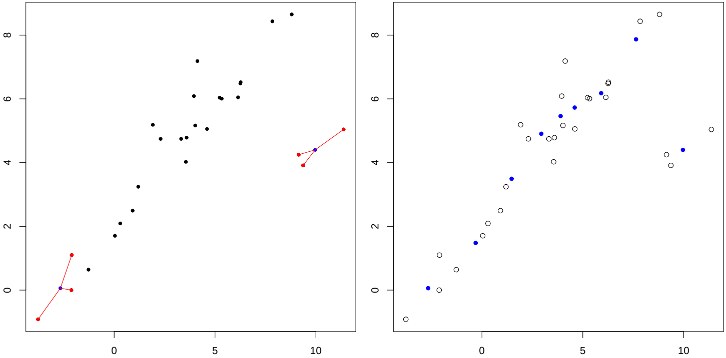
Example MDAV, 2-dim



Example MDAV, 2-dim



Example MDAV, 2-dim



Example microaggregation with R

```r
sdc <- microaggregation(sdc, aggr = 3, method="mdav")
print(sdc, "numrisk")
```

```
## Numerical key variables: expend, income, savings
##
## Disclosure risk (~100.00% in original data):
##   modified data: [0.00%; 93.58%]
##
## Current Information Loss in modified data (0.00% in orig
##   IL1: 543198.41
##   Difference of Eigenvalues: 4.530%
## ----------------------------------------------------------
```

Risk measure: use it only for comparison

Utility: we will use better measures later on

Univariate microaggregation

- The individual ranking method (univariate microaggregation) is not recommended to use, but often applied because of its simplicity.
- The method replaces values by its aggregates column by column independently.
- First, the first column is sorted and the index of sorting is memorized to be able to sort the values back in the original order. Then the first k values are replaced by their aggregate (usually the arithmetic mean), the next k values are replaced by their aggregate, and so on, until all values are aggregated from the first variable. The variable is then back-sorted.
- This procedure is then applied on the other variables independently.

Clearly destroys the multivariate structure of the data set.

Univariate microaggregation in R

```r
sdc <- undolast(sdc)
sdc <- microaggregation(sdc, aggr = 3, method="onedims")
print(sdc, "numrisk")
```

```
## Numerical key variables: expend, income, savings
##
## Disclosure risk (~100.00% in original data):
##   modified data: [0.00%; 100.00%]
##
## Current Information Loss in modified data (0.00% in orig
##   IL1: 444503.76
##   Difference of Eigenvalues: 4.450%
## ----------------------------------------------------------
```

Adding uncorrelated (additive) noise

Normal noise:

$$\mathbf{z}_j = \mathbf{x}_j + \epsilon_j \quad , \tag{1}$$

where vector $\mathbf{x}_j$ represents the original values of variable j, $\mathbf{z}_j$ represents the perturbed values of variable j and ϵ_j (uncorrelated noise, or white noise) denotes normally distributed errors with $\epsilon_j \sim N(0, c \cdot s_{x_j})$ with c a constant and s the standard deviation, $Cov(\epsilon_l, \epsilon_k) = 0$ for all $k \neq l$.

Uniform noise: ...

Mulitplicative noise: ...

Adding correlated noise

Often the better choice than uncorrelated noise, because the multivariate structure will not be completely changed.

- The difference to the uncorrelated noise method is that the covariance matrix of the errors is now designed to be proportional to the covariance of the original data, i.e. $\epsilon \sim N(0, \Sigma_\epsilon = c\Sigma_{\mathbf{X}})$.
- There are several variants of methods available how to achieve this (we will not go into details here)

Random orthogonal matrix masking

Multiplicative correlated noise

ROMM (**R**andom **O**rthogonal **M**atrix **M**asking):

perturbed data are obtained by

$$\mathbf{Z} = \mathbf{A}\mathbf{X}$$

- whereby $\mathbf{A}$ is randomly generated and
- fulfils $\mathbf{A}^{-1} = \mathbf{A}^{T}$ (orthogonality condition).

Adding uncorrelated noise in R

```r
sdc <- undolast(sdc)
sdc <- addNoise(sdc, noise = 5, method="additive")
print(sdc, "numrisk")
```

```
## Numerical key variables: expend, income, savings
##
## Disclosure risk (~100.00% in original data):
##   modified data: [0.00%; 47.55%]
##
## Current Information Loss in modified data (0.00% in orig
##   IL1: 575665.71
##   Difference of Eigenvalues: 4.460%
## --------------------------------------------------------
```

Adding correlated noise in R

```
sdc <- undolast(sdc)
sdc <- addNoise(sdc, noise = 5, method="correlated2")
print(sdc, "numrisk")
```

```
## Numerical key variables: expend, income, savings
##
## Disclosure risk (~100.00% in original data):
##   modified data: [0.00%; 14.69%]
##
## Current Information Loss in modified data (0.00% in orig
##   IL1: 790729.59
##   Difference of Eigenvalues: 4.970%
## ----------------------------------------------------------
```

See illustrative figures on tollerance ellipses in the book.

Multiplicative correlated noise (ROMM) in R

Loooong computation time. . .

```
sdc <- undolast(sdc)
sdc <- addNoise(sdc, noise = 5, method="ROMM")
```

Shuffling

Sketch outline of the method:

Regression model: continuous key variables as a response, other variables for predictors

- Swap values based on ranks of expected values and ranks of original values
- Rank correlations are preserved
- We do not want to go into more details on shuffling, because
  - especially outliers are not polluted.
  - if the model is weak, the multivariate dependencies are changed a lot (even rank correlation preserves)
  - a perfect model results in no perturbation

Shuffling in R

```r
sdc <- undolast(sdc)
sdc <- shuffle(sdc,
        form=savings+expend ~ urbrur+walls+water)
print(sdc, "numrisk")
```

```
## Numerical key variables: expend, income, savings
##
## Disclosure risk (~100.00% in original data):
##   modified data: [0.00%; 0.15%]
##
## Current Information Loss in modified data (0.00% in orig
##   IL1: 5218170.02
##   Difference of Eigenvalues: 3.020%
## ---------------------------------------------------------
```

We should take more care to spedify a good model.

Conclusion so far

What we learned so far

- Legal background on SDC (from an applied statistics perspective)
- We know methods to specify the discosure risk
- We are able to apply anonymisation methods

Conclusion so far

What we learned so far

- Legal background on SDC (from an applied statistics perspective)
- We know methods to specify the discosure risk
- We are able to apply anonymisation methods

What is missing?

- How to evaluate the utility of anonymized data?
- Choice of anonymization methods depends on the SDC problem and data set. Is there a standardized way of applying methods?

Conclusion so far

What we learned so far

- Legal background on SDC (from an applied statistics perspective)
- We know methods to specify the discosure risk
- We are able to apply anonymisation methods

What is missing?

- How to evaluate the utility of anonymized data?
- Choice of anonymization methods depends on the SDC problem and data set. Is there a standardized way of applying methods?

What other topics are missing so far?

- Synthetic data simulation $\rightarrow$ if time, we will disuss it during the lecture, otherwise see presentation on Wednesday
- SDC for tabular data $\rightarrow$ presentation on Wednesday