Statistical Disclosure Control for Microdata Using the R-Package sdcMicro

Article *in* Transactions on Data Privacy · August 2008

Source: DBLP

CITATIONS
85

READS
634

1 author:

Matthias Templ
University of Applied Sciences and Arts Northwestern Switzerland
345 PUBLICATIONS **4,380** CITATIONS

TRANSACTIONS ON DATA PRIVACY 1 (2008) 67–85

Statistical Disclosure Control for Microdata Using the R-Package sdcMicro

Matthias Templ[*,**]

[*]Department of Methodology, Statistics Austria, Guglgasse 13, 1110 Vienna, Austria. [**]Department of Statistics and Probability Theory, Vienna University of Technology, Wieder Hauptstr. 8-10, 1040 Vienna, Austria.

E-mail: matthias.templ@statistik.gv.at, templ@tuwien.ac.at

Abstract. The demand for high quality microdata for analytical purposes has grown rapidly among researchers and the public over the last few years. In order to respect existing laws on data privacy and to be able to provide microdata to researchers and the public, statistical institutes, agencies and other institutions may provide masked data. Using our flexible software tools with which one can apply protection methods in an exploratory manner, it is possible to generate high quality confidential (micro-)data.

In this paper we present highly flexible and easy to use software for the generation of anonymized microdata and give insights into the implementation and the design of the **R**-Package *sdcMicro*. **R** is a highly extendable system for statistical computing and graphics, distributed over the net. *sdcMicro* contains almost all popular methods for the anonymization of both categorical and continuous variables. Furthermore, several new methods have been implemented. The package can also be used for the comparison of methods and for measuring the information loss and disclosure risk of the masked data.

Keywords. Statistical Disclosure Control, Microdata, Software Development, R.

1 Introduction

Nowadays a number of different concepts exist to make confidential data accessible to researchers and users. A short outline of these concepts is given below in order to point out the need for data masking and the need for flexible software for data masking.

1.1 Remote Access

The most flexible way to provide microdata may be by installing *remote access* facilities where researchers can have a look at the data using a secure connection. They can analyse the data and choose a suitable model, but it is not possible for them to download it. Implementations can be found in [15] or [2]. Unfortunately, in some countries remote access can only be partially applied, depending on the discipline from which the data originates. This is due to different legacy for data coming from different disciplines. In Austria, for example, only microdata masking or *remote execution* is applicable for official statistics because of the legal situation which prohibits viewing "original" data.

1.2 Remote Execution, Model Servers and Automatic Perturbation of Outputs

When remote execution is applied, it is often necessary to generate synthetic data. Researchers can build and try out their methods using these synthetic data. Only in the final stage these methods are applied to the original data by the data holders. At this stage the output must be checked to detect and to avoid confidentiality disclosure. Without such synthetic data the data holders would have to check the output at every request which is very time-consuming and expensive. In addition to that, output checking is a highly sophisticated process in which the data holders may never be able to be sure if certain results lead to disclosure or not. Furthermore, data holders might never be able to understand all methods (for example available in more than 1500 **R**-packages) which might be applied to the data. Therefore, output checking may not be feasible at all.

Something between remote access and remote execution are the so-called *confidential preserving model servers* ([31]). Users are able to apply models to data which cannot be seen. Graphical summaries and parameter estimates of the data or of the resulting models are often perturbed (see e.g. in [13]) and especially outliers must be treated in a special way. Outlier detection itself is essential in many practical applications and may hardly be supported within this approach. Data almost always include outliers which can only be detected by robust procedures, i.e. to apply the whole range of diagnostic tools on robust estimates. Naturally, such robust estimates cannot be applied since the re-identification of outlying observations may be successful and the perturbation of outliers will make it impossible to select a model and a suitable method. On the other hand, the perturbation of summaries of non-smooth statistics - the approach of [12] - leads to serious problems. More details on this problem can be found in [34]. Another drawback is that only few methods can be implemented in such model servers within a reasonable time because every method must be adapted to avoid disclosure results when the users are applying these methods.

1.3 Data Masking

Our aim is to provide access to masked data since remote access cannot be applied due to legal restrictions in many cases and remote execution is too cost-intensive and time-consuming. We want to provide data which can be analysed by the researchers with their own methods and using their own software.

A popular software package, which has implemented some methods for masking data, is μ-Argus [17]. This software provides a graphical user interface (GUI) which may help users with little experience with statistical software to produce safe microdata sets.

In this paper we show that data masking can be easily performed with the newly-developed and flexible software package *sdcMicro* by minimizing information loss and the risk of re-identification. This software can be downloaded from the comprehensive **R** archive network on *http://cran.r-project.org* ([35]).

In addition to that, we also concentrate on the design of this new package for microdata protection and illustrate the open source philosophy of this project.

The outline of this contribution is as follows:
In chapter 3 we motivate why we have implemented methods for SDC in **R**. Chapter 4 briefly describes the methods which we have implemented in our package. We distinguish between methods for categorical and continuous scaled variables, followed by specific de-

sign issues of *sdcMicro*. In chapter 5 we show a practical application on real data in order to demonstrate the usage of the package. In chapter 6 we describe the open-source philosophy of our package and the benefits of such an open-source project related to SDC. Finally, we give a conclusion pointing out the capacity of package *sdcMicro*.

2 Using R for SDC

R [27] is an open source high-level statistical computing environment subjected to the General Public License and therefore freely available and extendable. Furthermore, **R** has become the standard statistical software and thousands of people are involved in the development of **R** both at universities and companies. More than 1400 add-on packages have been created over the last years.

We will now discuss and present the usefulness of **R** with respect to SDC.

2.1 Reproducibility and Interactivity

The most effective way of masking microdata is by doing the anonymization steps in an exploratory and in some sense iterative way. It is possible to apply various methods on various variables with different parameters resulting in different effects on the data while looking for sufficient anonymization of the data with respect to low information loss and low re-identification risk. Therefore, we must be able to reproduce any step of the anonymization process easily. This is possible by running a script which includes all the necessary commands. It is then easy to change and adapt the script, to run it and to get the updated results "in real time". Of course, this is fulfilled by many software tools but the real advantage of using **R** is that we can interactively "play" with the data, i.e. access all the objects in the workspace of **R** at any time and we can change, display or apply operations interactively on these objects on the fly. This is very useful during the anonymization process and is a quite different concept than writing a "batch file" which can then be executed.

2.2 Graphical Excellence

Powerful easy-to-grasp visualization tools display the effects of the anonymization on the data not only in order to compare the masked with the original data. For an effective exploratory approach during data anonymization such visualization tools are very helpful during the anonymization procedure because the effects of methods or parameter adjustments can be seen interactively. It is often more informative to look at some (comparison) plots to get a feeling about the information loss when perturbing the data, and to explore graphically distributions and the multivariate structure of the masked data and the original data instead of computing certain measures of information loss (see e.g. in [33] and the *scdMicro* package manual given in [35]).

2.3 Data Formats

Since numerous different data formats are handled by users it is very convenient to have the opportunity to import and export data formats from data base software like `MySQL`, `ODBC` database access, etc., statistical software like `SPSS`, `SAS`, `Stata`, "`Excel`" and data from other formats like `ASCII`, fixed format files and many more, supported within the **R** data import and export facilities (for more information see [28]).

2.4 Batch Mode and Platform Independence

Sometimes it is also useful to run the anonymization tools via batch mode which can be easily derived with the package *sdcMicro* [35]. It is also very important to provide software for SDC which is platform independent and works on all common operating systems.

2.5 Explorative Use

When perturbing categorical key variables - the variables on which an intruder may (partially) know and which may be used for re-identification by cross tabulation - recoding and local suppression come to mind. On the one hand the SDC-specialists try to recode as few as possible variables, and on the other hand as few as possible local suppressions should be made. Naturally, it is a highly exploratory process to find a way to recode only a subset of variables in a specific way in order to require only few suppressions, and at the same time to guarantee low individual risk and/or k-anonymity ([32]) [1]. As mentioned before, **R** is very well suited for interactive playing with the data and for trying out many different opportunities in "real time".
When perturbing numerical data the optimal perturbation method strongly depends on the multivariate structure of the data. Therefore, a flexible tool with which one can try out various methods and easily compare the methods is necessary. Additional flexibility may be provided by giving the user the opportunity to include his own functions for SDC.

2.6 Documentation

The provided **R** functions may be self-exploratory because the interested user can easily understand the method by looking at the source code. However, additional documentation is provided in a standardized way which includes the description of the method (or provides references, in case a detailed description is out of reach), the description of the function arguments, the description of the output as well as self-explaining executable examples. Furthermore, more general documentation with practical examples is available as a package vignette explaining parts of the package in a more informal way than the strict format of the usual **R** help files does by showing possible interactions of functions in the package with the help of practical examples.

All these topics are supported when using the software system **R**. In addition to that, dynamical reports can be used for documenting the anonymization process by using **R** in combination with `Sweave` ([21]). Hereby, the **R**-code for anonymization is directly placed in LaTeX files which are compiled using `Sweave`. Reports can be generated very effectively and quickly for particular steps in the production process, or to provide some fully-documented reports including graphics and **R** output for different sets of parameters. Using a fully-developed, object-oriented, open-source programming language is a great advantage for many reasons which are outlined in the following chapter. In addition to that, very large data sets can be processed in low computation time with this package.

[1] If the frequency count of every observation is greater than k. The frequency count for an observation is defined as the number of observations featuring the same values with respect to the key variables.

3 Objectives in the Design of Package *sdcMicro*

The advantages of using an object-oriented programming language become apparent in the *scdMicro* package. However, **R** is not a pure class-oriented programming language. It can be seen as a function- and class-oriented programming language which allows for a greater programming flexibility.

In **R** everything is an object and every object belongs to a specific class. The class of an object determines how it will be treated, and generic functions perform either a task or an action on its arguments according to the class of the argument itself (see e.g. in [29]). The class mechanism offers a facility for designing and writing functions for special purposes to the programmers which is extensively used in *sdcMicro*. Nearly all functions, e.g. the one for the individual risk methodology or the frequency calculation, produce objects from certain classes. Different *print*, *summary* and *plot* methods are provided for each of these objects depending on their class. `plot(ir1)` produces a completely different result than `plot(fc1)` assuming that the objects `ir1` and `fc1` are objects from different classes, i.e. resulting from different functions in package *sdcMicro*.
This object-oriented approach allows simple use of the package to any user, independently of the proficiency in **R**. Furthermore, users can try out methods with several different parameters and they can easily compare the methods with the implemented summary and plot methods.

Note that no metadata management needs to be done by the user, even not after importing the data into **R**. You can apply the methods directly on your data sets or on objects from certain classes. The only thing that has to be done is to determine which of the variables should be considered as the key variables, weight vector and as the confidential numerical variables.
An online documentation is included in the package containing all explanations on all input and output parameters of every function. Furthermore, various examples are included for each of the functions. All the examples can be easily executed by the users. Since *sdcMicro* is checked automatically every day from the CRAN server it is guaranteed that these examples work. Furthermore, additional checks are executed. E.g. the consistency of the documentation and the functions can be guaranteed.

To be able to deal with large data sets, expensive computational calculation steps are implemented in `C` and included in **R** via the `R/C++` interface. The estimation of the frequency counts for the population is one of the most critical calculation steps regarding the computation time. For example, Figure (1) shows the calculation time for frequency counts using the relatively small μ-Argus test data set with only 4000 observations (available within the anonymization software μ-Argus, see [18], version 4.1.0). It is carried out by a three-year old personal computer[2] running the Windows XP operating system. In order to estimate population frequency counts it is necessary to program with nested loops which is computationally expensive. It is therefore not feasable to run these estimations on large data sets using traditional `for`-loops in **R** because of possible long running times and required memory. Even when trying to avoid such `for`-loops in **R** using advanced concepts for data manipulation (using function `apply`, for example), the running time and the required memory is still a problem. Only when using advanced concepts of data manipulation in **R** for which the underlying functions are written in `C` or `Fortran` it is possible to calculate the frequency counts with up to 6 key variables in an adequate time. This is not possible

[2] Intel x86 based system with 3Ghz and 1 GB memory

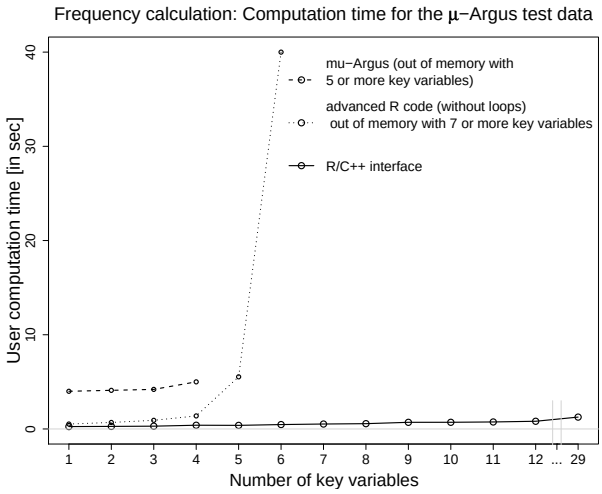

Figure 1: Computation time for calculating the frequency counts for the μ-Argus test data set with 4000 observations.

for the μ-Argus system which runs out of memory with 5 or more key variables. The developed `R/C` can also handle a enormous number of key variables[3] in a reasonable time, even when doing the calculation on much larger data sets than the μ-Argus test data set. In Figure (1) you can easily see that the computation time is always less than 1 second for this small data set, and it remains low for much larger data sets.

4 Implemented Methods and Data

Figure 2 shows the main structure of package *sdcMicro*. After reading the data (from a certain format) into **R** the user is able to protect the categorical key variables and/or to protect the confidential numerical variables. In addition to that, several functions are provided for the generation of synthetic data and for the measurement of the disclosure risk and the data utility for numerical data.

4.1 Methods for Masking Categorical Key Variables

Function `freqCalc()` calculates the frequency count for each observation as well as the frequency counts with respect to the sampling weights (details and a illustrative example can be found in [11]). Function `freqCalc()` returns an object of class *freqcalc*. For objects of this class a print and a summary method is provided (see Figure 2). Objects of class *freqCalc* can be used as input parameters to the functions `globalRecode()` and `indivRisk()`. `globalRecode()` provides some functionality for recoding variables. `indivRisk()` estimates the individual risk for re-identification as implemented in μ-Argus (see e.g. [11]) and produces objects of class *indivRisk* for which a plot and a print methods are implemented. Concerning the risk calculation, the uncertainty on the frequency counts of the population is accounted for in a Bayesian fashion by assuming that the population frequency given

[3] to choose such a large number of key variables is, of course, not always meaningful.

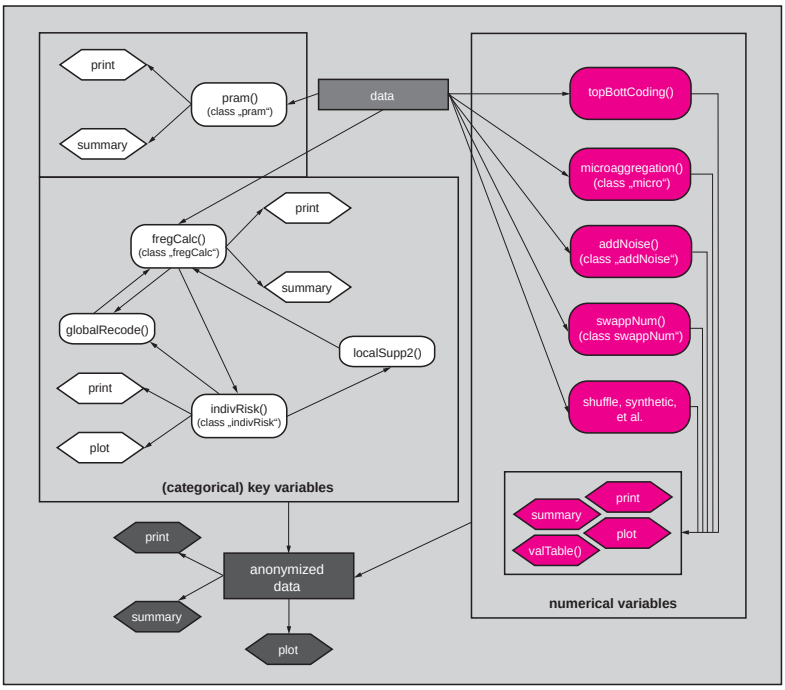

Figure 2: Survey on certain procedures in package *sdcMicro* and their relationship.

the sample frequency is negative binomial distributed. The objects of class *indivRisk* can be used as an input to the implemented local suppression functions. The local suppression function `localSupp2()` searches automatically for a quasi optimal solution to find a minimum amount of suppression. Additionally, the user may set weights for each variable to indicate which variables are less critical for suppression or which are more critical (the weight can be set to 0 for those variables which should not contain any suppressions and low weight for those which should only contain few suppressions). `localSupp2wrapper()` ensures to reach k-anonymity and low risk of re-identification.

It is recommended to use functions `freqCalc()`, `indivRisk()` and `globalRecode()` in an exploratory and sometimes in an iterative way. It takes just one look to know how many suppressions are necessary to achieve k-anonymity and/or low individual risk for different global recording settings. In addition to that, one can simply reproduce any previous step of the anonymization. So, one can try out different global recodings of various variables and evaluate its impact on information loss. Moreover, the frequency count calculation procedure can deal with missing values in the data.

Function `pram()` provides the PRAM methodology ([20]) and produces objects from class *pram*. When applying PRAM, values of categorical variables may change to different values, according to a pre-defined probability mechanism which is given in form of a specific transition matrix. A print method and a summary method are provided for objects of this class.

4.2 Methods for Masking Numerical Variables

The anonymization of numerical variables is often necessary to avoid a successful attack by record linkage methods.

Many different methods are implemented for masking numerical variables. For example there are more than 10 methods for microaggregation (*mdav* ([9]), *rmd* (which is an extension to *mdav* proposed in [34]), *individual ranking*, projection methods, and many more (see the references [33], [1], [10], [8], [7], [9]) in which the values of similar observations are aggregated by an "average" (which is often the arithmetic mean). In addition to that, five methods for adding noise are provided, namely, *adding additive noise* (the values of each variable are perturbed with a certain noise, see e.g. in [3]), *adding correlated noise* (the values of each variable are perturbed with a certain noise so that the covariance of these variables will be preserved, see [3]), *ROMM* (the perturbed data are obtained by $Y = AX$, whereas A is randomly generated and fulfills $A^{-1} = A^T$. To obtain an orthogonal matrix as described in [38], the Gram-Schmidt procedure was chosen in package *sdcMicro*) as well as method *outdect* which evaluates the "outlyingness" of each observation and adds noise depending on the "outlyingness" of an observation (see e.g. in [33]).

We also implemented numerical rank swapping ([6]) which is a method based on sorting a variable by its values (ranking). Each ranked value is then swapped with another ranked value which had been randomly chosen within a restricted range, i.e. the rank of two swapped values cannot differ by more than p percent of the total number of values.

Methods `gadp()` and `shuffle()` are also implemented in *sdcMicro*). These methods were originally proposed by [25] where the model $Y = S\beta + \epsilon$ is the object of interest. Y are perturbed variables, S are non-confidential variables and $\epsilon \sim MVN(0, \Sigma_{XX} - \Sigma_{XS}(\Sigma_{SS})^{-1}\Sigma_{SX})$. The regression coefficients β are estimated by $\hat{\beta} = (S'S)^{-1}S'X$ using X, the confidential variables. Shuffling is then done by replacing the rank ordered values of Y (generated by GADP) with the rank ordered values of X. Please note, that these methods are not officially available on CRAN because these methods are under US-patent. Since

these methods will only work reasonably when the data does not contain outliers, a robust version of these methods was proposed by [37] where robust Mahanlanobis distances are used to identify outliers and robust regression methods are applied. Since almost all data from official statistics contains outliers, it is highly recommended to use the robust version of shuffling.

There is also a method for fast generation of synthetic data included (details can be found in [22]) with which multivariate normal distributed data can be generated with respect to the covariance of the original data.

4.3 Data Sets Available in the Package

Several test data sets are included in the package. Some very small test data sets which were used by other authors for demonstration in the past are provided. Both the test data set from μ-ARGUS [18] and the test data sets from the CASC project (Tarragona, Census and EIA data sets, see e.g. [14]) can be used.

4.4 New Methods

Several new algorithms for microaggregation are included and are also available in package *sdcMicro*. A simple approach is to cluster the data first and then to sort the data in each group by the most influential variables in the groups, or to sort the observations in each group by the first robust principal component obtained by projection pursuit (method `clustpppca` in package *sdcMicro*, see [33] for details).

We have proposed a new algorithm for microaggregation called RMDM (**R**obust **M**ahalanobis **D**istance based **M**icroaggregation) where MDAV [9] is adapted in the following way:

1. Compute the robust center of the data. This can be the L_1-median or the coordinate-wise median.
2. Consider the most distant observation x_r to the robust center using robust Mahalanobis distances. The MCD-Estimator ([30]) can be used to calculate the robust covariance matrix which is needed for the calculation of the robust Mahalanobis distances.
3. Find the most distant observation x_s of x_r using Euclidean distance.
4. Choose $k-1$-nearest neighbors from x_r and also from x_s. Aggregate x_r and its $k-1$ nearest neighbors with an average as well as x_s with their $k-1$ nearest neighbors. The average can be the arithmetic mean but also a robust measure of location.
5. Take the previous dataset minus the aggregated data from the last step as a new dataset and continue with (1.) until all observations are microaggregated.

If the number of observations is not equal to a multiple of $2k$ then the algorithm aggregates the last remaining unaggregated $2k-h,\ h \in \{1,\ldots,k-1\}$ observations with the chosen average (see also in [9]). This proposed algorithm is more natural than the original MDAV algorithm since we deal with multivariate data taking the covariance structure of the data into account.

New methods for distance-based disclosure risk estimation for numerical data are proposed by [36] which consider again the "outlyingness" of observations. These methods are

implemented as well as the methods proposed by [23]. The main difference to the methods of [23] is that depending on the "outlyingness" of an observation the disclosure risk interval around the original data or the masked data increases or decreases. This is a more natural approach since observations in the center of the data cloud generally bear a lower risk of re-identification than observations which are outliers.

A comparison of almost all methods can be found in [37] and can easily be derived by using package *sdcMicro*.

An overview of the methods implemented in the package is also given in Table 1 together with information about references and names of the methods (functions) in the package. There is also information about the computation time by using the μ-Argus data set with the same key variables as used in section 5. The rather subjective information of the complexity of the implementation may be informative for many researchers in this field.

5 A Small Tour in *sdcMicro*

To stay within the limit of pages only a small tour in *sdcMicro* can be done excluding most of the graphical results and some steps of recoding variables. Comments in the code are marked with #, the output from **R** with `R>`. For further details please have a look at the examples and documentation which are included in package *sdcMicro*.

Given the fact that you have already installed **R** one possibility of installing the *sdcMicro* package plus all required packages from a CRAN server is to type the following command in **R**

```
install.packages("sdcMicro", depend=TRUE)
```

Now you can load both the package and your data using the powerful **R** import and export facilities. For the sake of simplicity we use the μ-Argus test data and print the first eight columns (argument before `]` in the code below) of the first four observations (argument after `[`) of the data [4].

```
library(sdcMicro); data(free1); xtable(free1[1:4, 1:8])
```

	REGION	SEX	AGE	MARSTAT	KINDPERS	NUMYOUNG	NUMOLD
1	36	1	43	4	3	0	0
2	36	1	27	4	3	0	0
3	36	1	46	4	1	0	0
4	36	1	27	4	1	0	0

Please note that you can simply use your own more meaningful data. We just use this data set (partially) because many SDC-specialists are familiar with this data set and have tried out methods using this data within the μ-Argus software.

In the following code segment the frequency counts are calculated as described in [5] and allocated to object `fr1` which is now automatically of class *freqCalc*. Several methods are available for this class. Object `fr1` is then used as an input for the individual risk computation. A new object `ir1` of class `indivRisk` will be produced. Several methods are available for this class as well. For example, function `plot` automatically knows which plot method must be used for an object of class *indivRisk*. Figure (4) is generated by this plot method. The implementation of this plot method for individual risk methods is quite similar to the one in μ-Argus.

[4] `xtable()` produces a LaTeX-styled print output.

Table 1: An overview of the most important methods which are implemented in *sdcMicro*. Note that for almost all of these methods *print*, *summary* and *plot* methods are implemented (and not listed here). Only those references to papers which have been used for the implementation of the methods in *sdcMicro* are given.

method	ref.	funtion name in *sdcMicro*	parameter in the function	implem.	ctime
PRAM	[20]	pram	–	easy	0.09
frequency calculation	[5]	freqCalc	–	depends	0.516
indivRisk	[11]	indivRisk	–	medium	0.06
(optimal) local suppression	–	localSupp2-wrapper	–	hard	484
adding additive noise	[4]	addNoise	additive	very easy	0.01
correlated noise	[4]	addNoise	correlated	easy	0.01
correlated noise 2	[19], [16]	addNoise	correlated2	easy	0.01
restricted correlated noise	[3]	addNoise	restr	easy	0.01
ROMM	[38]	addNoise	ROMM	easy	> 2000
adding noise based on multivariate outlier detection	[33], here	addNoise	outdect	hard	0.18
rank swapping	[6]	swappNum	–	very easy	0.88
microaggregation (ma), individual ranking	[8]	microagg.	onedims	easy	0.16
ma, sorting on first principal component	–	microagg.	pca	easy	0.1
ma, sorting with projection pursuit pca on clustered data	[33], [34]	microagg.	clustpppca	hard	1.54
ma, mdav	[16]	microagg.	mdav	easy	> 200
ma, multivariate microaggregation based on robust Mahalanobis distances	[34]	microagg.	rmd	hard	0.75
gadp	[25], [26]	shuffle	–	very easy	0.16
shuffling	[26]	shuffle	–	very easy	0.19
shuffling based on cgadp	[26]	shuffle2	–	easy	0.19
robust gadp and robust shuffling	[37]	robGadp, robShuffle	–	medium	0.4
synthetic, cholesky decomposition	[22]	gendat	–	easy	0.01

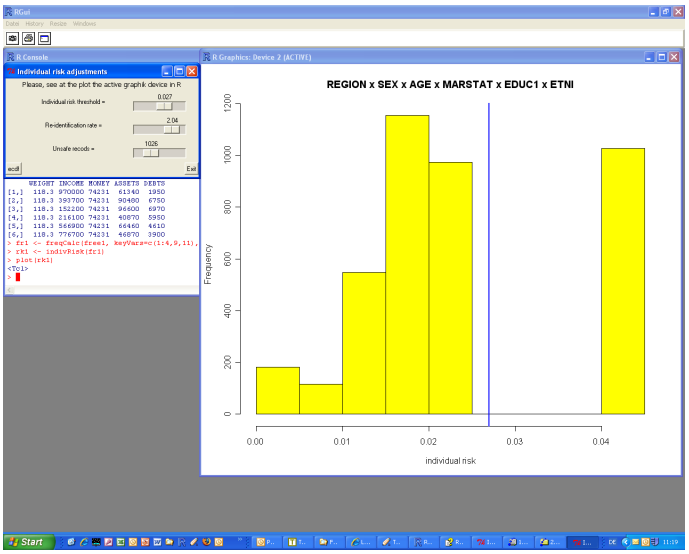

Figure 3: Individual risk. In the upper region of the figure you will see a helpful slider which is directly linked with the graphics.

Instead of providing metadata (which can be quite cumbersome) we only use the data itself. The user must only provide the information on which variables should be treated as the key variables and which one includes the sampling weights (if there is such a variable in the data). Since we neither have a detailed variable description nor know much about the data (in practice this lack of information does not exists, because the data holder does have such information) we suppose that variables *REGION*, *SEX*, *AGE*, *MARSTAT*, *ETNI* and *EDUC1* may be treated as key variables because a data intruder might have some information about some observations of these variables. Please note that it is not possible to calculate these frequency counts in μ-Argus (it fails with an error message which reports that the program ran out of memory). Figure 3 shows an interactive plot of the individual risk similar to μ-Argus. Here, a histogram of the indidividual risk is shown but the empirical cummulative distribution function of the risk can be also plotted, which might be more informative.

```
kv <- which(colnames(free1) %in% c("REGION", "SEX", "MARSTAT", "ETNI", "EDUC1"))
fr1 <- freqCalc(free1, keyVars=kv, w=30)
rk1 <- indivRisk(fr1)
class(rk1)                    ## class of object rk1
R> [1] "indivRisk"
methods(class = indivRisk)    ## methods available for objects of this class
R> [1] plot.indivRisk   print.indivRisk
plot(rk1)
```

The script shown above can easily be adapted (exclude the **R** results), e.g. by adding the function `globalRecode()`. `globalRecode()` recodes several categories of a variable into less categories or discretises a numerical variable. So, this function checks the class of a variable and recodes the variables based on its class. The user is still able to manipulate the data in an exploratory way. You might try to recode a variable in order to observe the influence of your recoding on the frequency count calculation and the individual risk computation, but also to find out how many suppression will be needed when changing

the coding of some key variables. Note that you can easily reproduce all your steps either by running a part of your script or the whole script.

We now show just the code for the final recoding, namely recoding of `region` into 3 equally sized categories, recoding of *age* into 5 specific categories, recoding of education into 2 categories and recoding of ethnicity into 2 categories. This results in 54 observations which are unique in the sample (instead of 3702 without recoding). Since we do not have any information about these categories of this test data set, we only show a number of recodings which might not be meaningful. Naturally, the user has the information which category consits of which group, i.e. if education is equal to 9 means university degree or primary school, for example. With such information one may join the most similar categories into new categories. We are always able to check the effects of the recoding, e.g. showing how many observations are unique in the sample (`fk=1`).

```
x <- data.frame(free1)
attach(x)
x$REGION <- as.integer(globalRecode(REGION, breaks=3, method="equalAmount"))
x$AGE <- as.integer(globalRecode(AGE, breaks=c(14,30,45,55,74)) )
table(EDUC1)
   1    2    3    4    5    6    7    9
 781  437  620  268 1223  457  209    5
x$EDUC1 <- as.integer(globalRecode(EDUC1, breaks=c(0,4,9)))
table(ENTI)
   1    2    3    4    5    6    7    8    9
3677   23   19   13   73   79   55   10   51
x$ETNI <- as.integer(globalRecode(ETNI, breaks=c(0,1,9)))
fr2 <- freqCalc(x, keyVars=c(1:4,9,11), w=30)
fr2

  --------------------------
54 observation with fk=1
63 observation with fk=2
  --------------------------
rk2 <- indivRisk(fr1)
detach(x)
```

It is really easy to see the effects of different recodings in less than 0.5 seconds, for example, when allowing more regions (e.g. 5). Just type the information about the new breaks into this script and run the code resulting in

```
fr2

  --------------------------
107 observation with fk=1
110 observation with fk=2
  --------------------------
```

After minimising the re-identification risk you can apply local suppression (function `localSupp2()`) on object `indivf` and `fr1` to delete the last unsureness about risky observations and then make use of the implemented print and summary methods. But global recoding and local suppression may, of course, also be used in an alternated manner. With parameter *importance* in function *localSupp2Wrapper* weights to each key variable can be assigned. Variables with higher weights will be preferred for suppression. In our example we assume that suppressions in the last 3 key variables are more acceptable for the user than in the other key variables. We want to perform 3-anonymity.

```
ls1 <- localSupp2Wrapper(x,keyVars=kv,w=30,importance=c(0.2,0.2,0.3,1,1,1),kAnon=3)
ls1
  ----------------------
[1] "Total Suppressions in the key variables 280"
```

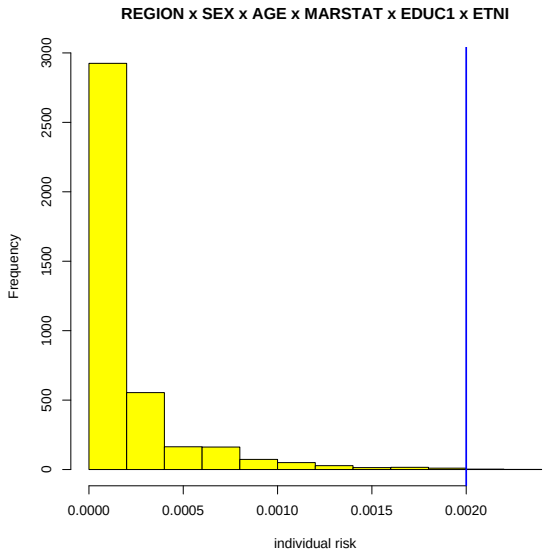

Figure 4: Individual risk after the anonymization of the μ-Argus test data set.

```
[1] "Number of suppressions in the key variables "

 4  0  0 202 54 20
 ------------
[1] "3-anonymity == TRUE"

 ----------------------
```

So, 4 values in *REGION* are supressed and 202, 54, and 20 values are suppressed in *MARSTAT*, *EDUC1* and in variable *ETNI* respectively.

Finally, one can check if observations with high individual risk remain and may suppress some additional values (for observations with a risk above a certain threshold) in order to reduce the risk (using function *localSupp*). This can easily be done by

```
fff <- freqCalc(ls1$xAnon, keyVars=c(1:4,9,11), w=30)
rk11 <- indivRisk(fff)
plot(rk11)
```

In addition to that, you can simply microaggregate numerical variables with more than 10 different methods or use another method described previously.

```
x[, 31:34] <- microaggregation(x[, 31:34], aggr=4, method="clustpppca")
```

There are a lot of comparison plot methods available to compare the perturbed data and the original data. You can easily compare the different methods. We show this on another

data set, the *Tarragona* data set[5], because the μ-Argus test data does include faked numerical variables which follow a multivariate uniform structure without correlations between the variables.

Evaluating different methods for the masking of numerical variables becomes easy by using function `valTable()` which estimates different measures of information loss and distance-based disclosure risk.

```
data(Tarragona)
v <- valTable(Tarragona, method=c("addNoise: additive", "addNoise: correlated2",
"addNoise: outdect","swappNum","onedims","pca","clustpppca","mdav","rmd","dataGen"))
xtable(v[, c(1,3,6,9,15,17,20)])   ## choose some measures and produce Table 2
```

Table 2: Comparison of different methods regarding a univariate measure of information loss (mean absolute error of medians), one multivariate measure of information loss (mean absolute error of correlations), one utility measure, the risk measure given in [23] and a new risk measure weighted with the robust Mahalanobis distance. Further measures were evaluated but not printed in this example.

	method	amedian	amad	acors	util1	risk0	risk2
1	addNoise: additive	0.23	0.16	0.18	9.61	0.04	0.99
2	addNoise: correlated2	3.57	5.88	1.20	93.96	0.00	1.00
3	addNoise: outdect	0.26	0.24	0.23	14.99	0.59	0.99
4	swappNum	0.13	0.18	1.28	386.62	0.00	0.99
5	onedims	0.03	0.04	0.01	18.63	0.80	0.99
6	pca	2.62	1.10	9.57	232.68	0.00	0.00
7	clustpppca	3.64	1.55	7.40	259.34	0.00	0.00
8	mdav	1.98	0.83	5.68	197.10	0.00	0.00
9	rmd	0.83	0.42	1.48	170.96	0.00	0.00
10	dataGen	27.53	78.48	5.89	1162.14	0.00	0.83

The first column of Table 2 represents a univariate measure of information loss (absolute deviations from the medians), the next column represents a multivariate measure of information loss (differences in spearman correlation coefficients) followed by a data utility measure and two distance-based disclosure risk measures. A detailed description of these measures can be found in the online-help files from package *sdcMicro* ([35]). One can easily compare the performace of these methods on the Tarragona data set. Method adding additive noise performs quite well but bears a high disclosure risk, for example. However, Table 2 shows only one configuration and naturally for each method some parameters can be changed so that the data utility increases and the disclosure risk decreases or vice versa. Methods `shuffle()` and `gadp()` are not included in Table 2 since it would be rather subjective which variables should be chosen to be the "non-confidential" ones to protect these variables. In many cases method RMDM (in package *sdcMicro* the method is named `rmd`) performs best since it results in relatively low disclosure risk and very low information loss.

Another method for categorical variables is PRAM which can easily be applied with function `pram()`. In the following, we perturbe variable `MARSTAT` form the μ-Argus test data set with the invariant PRAM methodology. A lot of information is stored in object

[5] http://neon.vb.cbs.nl/casc/testsets.html

MARSTATpram, e.g. the invariant transition matrix. Summary and print methods are provided as well. A selection of the output of the summary is given below.

```
MARSTATpram <- pram(free1[,"MARSTAT"])     ## with default parameters
summary(MARSTATpram)
  ----------------------
  original frequencies
    1    2    3    4
2547  162  171 1120

  ----------------------
  frequencies after perturb.:

    1    2    3    4
2571  160  178 1091

  ----------------------
   transition Frequency
1      1 --> 1      2448
2      1 --> 2        27
3      1 --> 3        28
4      1 --> 4        44
5      2 --> 1        33
6      2 --> 2       118
7      2 --> 3         4
8      2 --> 4         7
9      3 --> 1        20
10     3 --> 2         3
11     3 --> 3       130
...      ...         ...
```

6 Open Source Initiative

As mentioned above, the entire code is freely available and can be downloaded from *http://cran.r-project.org*. So you can learn from this code, change it yourself or develop the package *sdcMicro* further. Instead of keeping the developed code to yourself you are invited to contribute to this package. Every response and bug report will be helpful in achieving a higher quality of the package. Note that up to now the quality of the package was highly improved by comments and bug reports from many users from statistical offices and companies. *sdcMicro* is actively being developed and improved, and so it is highly recommended to use the latest version of the package.

Every function has its own author and the copyright of the function belongs to the author. This means that nobody can use your functions for a commercial software product. On the other hand the intellectual property is also ensured. Please note that the intellectual property must be guaranteed when using the functions for publications in journals (please reference when using *sdcMicro* or parts of the package). But at the same time all the functions are open-source and everybody can, and should, use them.

7 Conclusion

The main difference between *sdcMicro* and the popular software μ-Argus is the possibility to use SDC methods in an exploratory manner in *sdcMicro*. So, different parameter settings can be tried out during the data masking process and a detailed look at each step of the anonymization using implemented print, summary, plot, information loss and disclosure risk methods is at hand. Additionally, the whole functionality of **R** can be used. All the steps of the anonymization can easily be reproduced.

We have shown by anonymizing the μ-Argus test data set that the anonymization procedure becomes easy using **R**-package *sdcMicro*. Moreover, for your own data sets it might get even simpler because detailed information about the variables and its categories is at hand.

The potential of this package can become very high and the package has a realistic chance to become the most important implementation for SDC in microdata protection not only

because all methods from μ-Argus are available but also because it features several new developments. The response of users from all over the world is very positive and the package is already used in the production process (see e.g. [24]). The users are still satisfied with the command line interface, and so an implementation of a graphical user interface has not been developed yet. In addition to this package, the entire power of **R** can be used to boost the results. Everybody is invited to contribute to this package, especially in funded future research projects.

Acknowledgements

My thanks go to Bernhard Meindl for numerous suggestions and contributions to the code and to Alois Haslinger who supported the development of the package by giving me the necessary time for this work.

References

[1] N. Anwar. Micro-aggregation - the small aggregates method. In *Internal report*. Luxembourg: Eurostat, 1993.

[2] L. Borchsenius. New developments in the danish system for access to micro data. In *UN-ECE/EUROSTAT Worksession on Statistical Data Confidentiality, Geneva*, 2005.

[3] R. Brand. Microdata protection through noise addition. In *Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer*, pages 347–359, 2004.

[4] R. Brand and S. Giessing. Report on preparation of the data set and improvements on sullivans algorithm. Technical report, 2002.

[5] A. Capobianchi, S. Polettini, and M. Lucarelli. Strategy for the implementation of individual risk methodology into μ-ARGUS. Technical report, Report for the CASC project. No: 1.2-D1, 2001.

[6] T. Dalenius and S.P. Reiss. Data-swapping: A technique for disclosure control. In *Proceedings of the Section on Survey Research Methods*, volume 6, pages 73–85. American Statistical Association, 1982.

[7] D. Defays and Anwar M.N. Masking microdata using micro-aggregation. *Journal of Official Statistics*, 14(4):449–461, 1998.

[8] D. Defays and P. Nanopoulos. Panels of enterprises and confidentiality: the small aggregates method. In *Proceedings of the 1992 Symposium on Design and Analysis of Longitudinal Surveys*, pages 195–204. Statistics Canada, Ottawa, 1993.

[9] J. Domingo-Ferrer and J.M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Trans. on Knowledge and Data Engineering*, 14(1):189–201, 2002.

[10] M. Elliot, A. Hundepool, E.S. Nordholt, J-L. Tambay, and T. Wende. Glossary on statistical disclosure control, 2005.

[11] L. Franconi and S. Polettini. Individual risk estimation in μ-Argus: a review. In J. In: Domingo-Ferrer, editor, *Privacy in Statistical Databases, Lecture Notes in Computer Science*, pages 262–272. Springer, 2004.

[12] J. Heitzig. The 'jackkife' method: confidentiality protection for complex statistical analyses. In *Joint UNECE/Eurostat work session on statistical data confidentiality, Geneva, Switzerland*, 2005.

[13] J. Heitzig. Using the jackknife method to produce safe plots of microdata. In *Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer*, pages 139–151, 2006.

[14] A. Hundepool. The casc project. In *Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer*, pages 199–212, 2004.

[15] A. Hundepool and P-P. de Wolf. Onsite@home: Remote access at statistics netherlands. In *Monographs of official statistics, Work session on statistical data confidentiality*. Eurostat, Luxembourg, 2005.

[16] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, R. Lenz, J. Longhurst, E. Schulte-Nordholt, G. Seri, and P.-P. De Wolf. Handbook on statistical disclosure control version 1.01, 2007.

[17] A. Hundepool, A. Van deWetering, Ramaswamy R., L. Franconi, A. Capobianchi, P-P. DeWolf, J. Domingo-Ferrer, V. Torra, R. Brand, and S. Giessing. μ-argus version 3.2 software and users manual, 2005.

[18] A. Hundepool, A. Van deWetering, Ramaswamy R., L. Franconi, A. Capobianchi, P-P. DeWolf, J. Domingo-Ferrer, V. Torra, R. Brand, and S. Giessing. μ-argus version 4.1 software and user manual, 2006.

[19] J.J. Kim. A method for limiting disclosure in microdata based on random noise and transformation. In *Proceedings of the Section on Survey Research Methods*, pages 303–308. American Statistical Association, 1986.

[20] P. Kooiman, L. Willenbourg, and J. Gouweleeuw. A method for disclosure limitation of microdata. Technical report, Research paper 9705, Statistics Netherlands, Voorburg, 2002.

[21] Friedrich Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, December 2002.

[22] J.M. Mateo-Sanz, A. Martínez-Ballesté, and J. Domingo-Ferrer. Fast generation of accurate synthetic microdata. In J. In: Domingo-Ferrer, editor, *Privacy in Statistical Databases, Lecture Notes in Computer Science*, pages 298–306. Springer, 2004.

[23] J.M. Mateo-Sanz, F. Sebe, and J. Domingo-Ferrer. Outlier protection in continuous microdata masking. *Lecture Notes in Computer Science, Vol. Privacy in Statistical Databases, Springer Verlag*, 3050:201–215, 2004.

[24] B. Meindl and M. Templ. The anonymisation of the CVTS2 and income tax dataset. an approach using R-package sdcMicro. In *to appear in: Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality. Monographs of Official Statistics.*, 2007.

[25] K. Muralidhar, R. Parsa, and R. Sarathy. A general additive data perurbation method for database security. *Management Science*, 45:1399–1415, 1999.

[26] K. Muralidhar and R. Sarathy. Data shuffling- a new masking approach for numerical data. *Management Science*, 52(2):658–670, 2006.

[27] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[28] R Development Core Team. *R Import Export*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[29] R Development Core Team. *The R language definition*. R Foundation for Statistical Computing, Vienna, Austria, 2008.

[30] Van Driessen K. Rousseeuw P.J. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 1999.

[31] P. Steel and A. Reznek. Issues in designing a confidential preserving model server. In *Monographs of official statistics, Work session on statistical data confidentiality*. Eurostat, Luxembourg, 2005.

[32] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[33] M. Templ. Software development for SDC in R. In *Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer*, pages 347–359, 2006.

[34] M. Templ. sdcMicro: A new flexible R-package for the generation of anonymised microdata - design issues and new methods. In *to appear in: Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality. Monographs of Official Statistics.*, 2007.

[35] M. Templ. *sdcMicro. Manual and Package.* Statistics Austria and Vienna University of Technology, Vienna, Austria, 2007. *http://cran.r-project.org/src/contrib/Descriptions/sdcMicro.html*.

[36] M. Templ and B. Meindl. Robust statistics meets SDC: New disclosure risk measures for continuous microdata masking. In *Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer (submitted and in review)*, 2008.

[37] M. Templ and B. Meindl. Why shuffle when you can use robust statistics for SDC - a simulation study. In *Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer (submitted and in review)*, 2008.

[38] D. Ting, S. Fienberg, and M. Trottini. ROMM methodology for microdata release. In *Monographs of official statistics, Work session on statistical data confidentiality*. Eurostat, Luxembourg, 2005.