Journal of Statistical Software

October 2015, Volume 67, Issue 4. doi: 10.18637/jss.v067.i04

Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro

Matthias Templ
Vienna University of Technology

Alexander Kowarik
Statistics Austria

Bernhard Meindl
Statistics Austria

Abstract

The demand for data from surveys, censuses or registers containing sensible information on people or enterprises has increased significantly over the last years. However, before data can be provided to the public or to researchers, confidentiality has to be respected for any data set possibly containing sensible information about individual units. Confidentiality can be achieved by applying statistical disclosure control (SDC) methods to the data in order to decrease the disclosure risk of data.

The R package **sdcMicro** serves as an easy-to-handle, object-oriented S4 class implementation of SDC methods to evaluate and anonymize confidential micro-data sets. It includes all popular disclosure risk and perturbation methods. The package performs automated recalculation of frequency counts, individual and global risk measures, information loss and data utility statistics after each anonymization step. All methods are highly optimized in terms of computational costs to be able to work with large data sets. Reporting facilities that summarize the anonymization process can also be easily used by practitioners. We describe the package and demonstrate its functionality with a complex household survey test data set that has been distributed by the International Household Survey Network.

Keywords: confidentiality, micro-data, statistical disclosure control, R.

1. Introduction

Statistical disclosure control (SDC) is an emerging field of research. More and more data on persons and establishments are collected by statistical organizations and almost all of these data holds confidential information. The demand for micro-data for researchers is increasing since economic analysis or empirical analysis of the society is often only possible by investigating data containing detailed information. In addition, also the demand for complex micro-data for training purposes seems to increase. Furthermore, public-use and

Method \ Software	μ-**Argus** 4.2	**sdcMicro** 1.0.0	**sdcMicro** > 4.3.0	**sdcMicroGUI** > 1.1.0	IHSN
Frequency counts	✔	(✔)	✔	✔	✔
Individual risk	✔	(✔)	✔	✔	✔
Individual risk on households	✔		✔	✔	✔
l-diversity			✔	✔	✔
SUDA2			✔		✔
Global risk	✔		✔	✔	✔
Global risk with log-lin mod.			✔		
Recoding	✔	(✔)	✔	✔	(✔)
Local suppression	(✔)	(✔)	✔	✔	(✔)
Swapping	(✔)	(✔)	✔		✔
PRAM	✔	✔	✔	✔	✔
Adding correlated noise		✔	✔	✔	✔
Micro-aggregation	✔	(✔)	✔	✔	✔
Shuffling			✔	✔	
Utility measures	(✔)	✔	✔	✔	
GUI	(✔)			✔	
CLI		✔	✔		✔
Missing values	✔		✔	✔	✔
Cluster designs	✔		✔	✔	✔
Large data			✔	✔	(✔)
Reporting	✔		✔	✔	
Platform independent		✔	✔	✔	✔
Free and open-source		✔	✔	✔	✔

Table 1: List of methods supported by different statistical disclosure control software. Ticks are in brackets when only limited support is provided to a method. A comparison to version 1.0.0 of **sdcMicro** (released May 29, 2007; published in Templ 2008) is given to indicate the progress of the new complete reimplementation of the package.

scientific-use files are required by researchers to run complex simulation studies to compare methods. As examples for such simulation studies, the *BLUE-ETS*, *AMELI*, *DACSEIS* and *EUREDIT* research projects from the Framework Programmes for Research of the European Union can be named. As examples for releases of public-use data sets, we mention data sets from the U.S. Census Bureau including the Current Population Survey, the Survey of Income and Program Participation, the American Housing Survey, the Survey of Program Dynamics, the American Community Survey and the Consumer Expenditure Survey (see, e.g., Task Force on the Conference of European Statisticians 2007).

In any case, due to national laws on privacy, micro-data cannot be distributed to the public or to researchers whenever re-identification of persons or establishments is possible. Software packages are fundamental for the anonymization of data sets. Table 1 gives an overview on methods available in four software products – the μ-**Argus** software (Hundepool *et al.* 2008) from Statistics Netherlands, different versions and an extension of the R package **sdcMicro** (Templ, Kowarik, and Meindl 2015) and C++ code that was written by the International Household Survey Network (IHSN, see `http://www.ihsn.org/home/node/32`). For the sake

of completeness, a former version of **sdcMicro** (Templ 2008) is also listed as well as the point and click graphical user interface **sdcMicroGUI** (Kowarik, Templ, Meindl, and Fonteneau 2013) that serves as an easy-to-handle, highly interactive tool for users who want to use the **sdcMicro** package but are not familiar with the native R command line interface. **μ-Argus** is a general tool for the anonymization of micro-data that has been developed by Statistics Netherlands with support from Eurostat and the 5th Framework Programme for Research of the European Union. **sdcMicro** version $>$ 4.3.0 provides many more methods than **μ-Argus** and **sdcMicro** version 1.0.0, but major differences can also be found in the implementation. The main advantages of the proposed **sdcMicro** package consists of its user-friendly object-oriented application, the ease of importing data (in **μ-Argus** an import script which determines the hierarchical structure of data has to be written), its flexibility to use and the efficient implementation. We cannot compare computation speed of **μ-Argus** to **sdcMicro**, as methods cannot be applied using a command line interface, but we would like to point out that **μ-Argus** is not suitable for large data sets. It becomes slow and runs out-of-memory even with medium-sized data sets. This is also true for **sdcMicro**, version 1.0.0.

This contribution is structured in the following manner. First the basic concepts to anonymize micro-data are briefly introduced in Section 2 and possible challenges related to large complex data sets are mentioned. A work flow shows how and when methods may be applied during the whole anonymization process. The conceptional difference regarding the distribution of the variables is underlined. In Section 3 the main functions of **sdcMicro**, their general usage and the object-oriented approach using S4 classes (Section 3.2) is explained. The next section, Section 4, gives more details about supported SDC methods. It starts with disclosure risk methods, followed by anonymization methods and methods to measure data utility. The application of the related **sdcMicro** functions is shown for each method. More precisely, the basic methods on measuring the disclosure risk of micro-data are explained in Section 4.1. Afterwards, a brief description of anonymization methods is given (Section 4.2). The measurement of data utility is included in Section 4.3. The reporting facilities are presented in Section 5. These reports summarize the whole anonymization applied to a defined object. Section 6 concludes with a summary of the main advantages of the package.

2. Concepts to anonymize micro-data

A micro-data file is a data set that holds information collected on individual units like people, households or enterprises. In general, disclosure occurs when an intruder uses the released data to reveal previously unknown information about an individual.

2.1. Classification of variables

For each unit/observation, a set of variables is recorded and available in the data set. These variables can be classified into groups, which are not necessarily disjunctive:

Direct identifiers are variables that precisely identify statistical units. For example, social insurance numbers, names of companies or persons and addresses are direct identifiers.

Key variables are a set of variables that, when considered together, can be used to identify individual units. For example, it may be possible to identify individuals by using a combination of variables such as gender, age, region and occupation. Key variables are

also called implicit identifiers or quasi-identifiers. When discussing SDC methods, it is preferable to distinguish between categorical and continuous key variables based on the scale of the corresponding variables.

Non-confidential variables are variables that are not direct identifiers or key variables. For example, a variable containing information about the number of TVs in a household is non-confidential since this information is usually not available in any public data base and thus cannot be linked with auxiliary data.

Sensitive variables are used for specific methods such as l-diversity (see Section 4.1). Information on cancer may serve as an example of a sensitive variable.

Clustering variables are variables determining a hierarchical structure (e.g., households).

Stratification variables are variables determining the application of methods on data subsets. Classification variables, such as economic branches, often serve as stratification variables and specific methods are then applied to each economic branch separately.

The goal of anonymizing micro-data is to prevent confidential information from being assigned to a specific respondent. If linkage based on a number of identifiers is successful, the intruder has access to all of the information related to a specific corresponding unit in the released data. This means that a subset of critical variables can be exploited to disclose everything about a unit in the data set.

2.2. Challenges

Methods used in statistical disclosure control borrow techniques from other fields. Multivariate methods are used to modify or simulate continuous variables (see, e.g., Muralidhar and Sarathy 2006; Templ and Meindl 2008b) and to quantify information loss (see, e.g., Domingo-Ferrer and Torra 2001) while distribution-fitting methods are used to quantify disclosure risks (see, e.g., Skinner and Holmes 1998; Franconi and Polettini 2004; Rinott and Shlomo 2006). Statistical modeling methods form the basis of perturbation algorithms (see, e.g., Muralidhar and Sarathy 2006), to simulate data (see, e.g., Drechsler 2011; Alfons, Kraft, Templ, and Filzmoser 2011) and to quantify risks and information loss (see, e.g., Shlomo 2010). Optimization methods may be used to modify data with a minimum impact on data quality (see, e.g., Fischetti and Salazar-González 2000).

Large data sets present unique problems and challenges and place an even higher demand on computational efficiency. Handling missing values and structural zeros can be particularly challenging. Complex data structures, like those that are common in survey research, compound these challenges.

SDC techniques can be divided into three broad topics:

- measuring disclosure risk (see Section 4.1);
- application of SDC-methods to deal with disclosure risks (see Section 4.2);
- comparing original and modified data (information loss) (see Section 4.3).

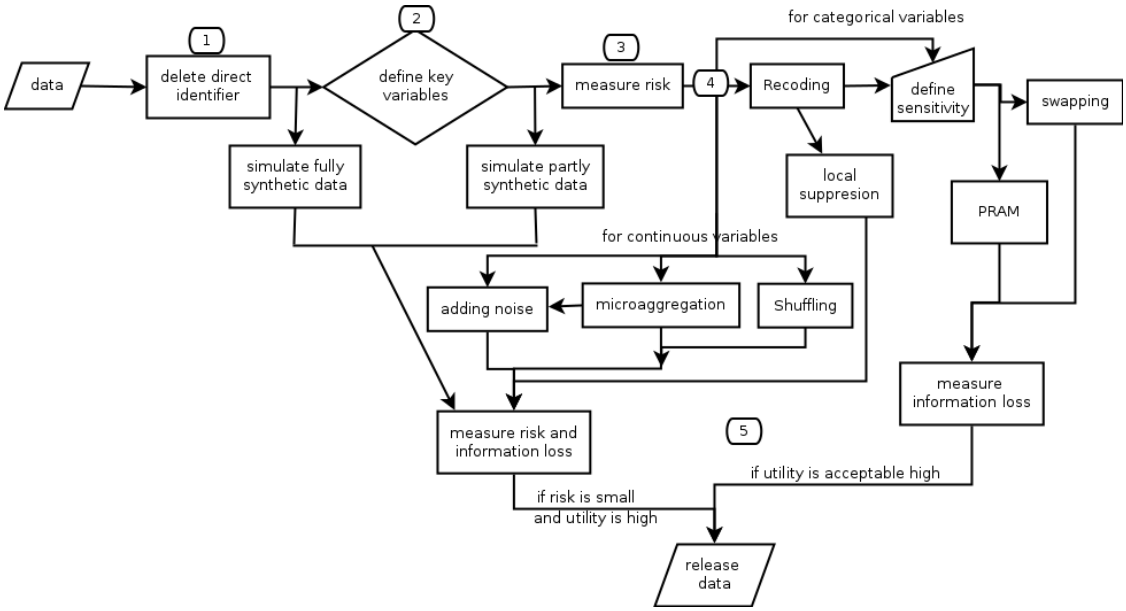

Figure 1: Possible work flow to anonymize micro-data using various SDC methods. The numbers 1–5 correspond to the enumerated explanations in Section 2.3.

2.3. Work flow

Figure 1 outlines common tasks, practices and steps required to obtain confidential data. The following steps to perturbate micro-data are included in Figure 1:

1. The first step is to remove all direct identification variables from the micro-data set.

2. Second, a set of key variables used for all further risk calculations has to be selected. This decision is subjective and involves discussions with subject matter specialists and interpretation of related national laws. See Templ, Kowarik, and Meindl (2014) for practical applications. For the simulation of fully synthetic data, choosing key variables is not necessary, see for example Alfons *et al.* (2011).

3. After key variables have been selected, disclosure risks of individual units are measured. This step includes the analysis of sample frequency counts as well as the application of probability methods to estimate individual re-identification risks by taking population frequencies into account.

4. Next, observations with high individual risks are modified. Techniques such as recoding and local suppression, recoding and swapping or PRAM (post randomization method Gouweleeuw, Kooiman, Willenborg, and De Wolf 1998) can be applied to categorical key variables. It is possible to apply PRAM or swapping without recoding of key variables beforehand. However, lower swapping rates might be possible if key variables are modified in advance. The decision as to which method to apply also depends on the structure of the key variables. In general, one can use recoding together with local suppression if the amount of unique combinations of key variables is low. PRAM should be used if the number of key variables is large and the number of unique combinations is high; for details, see Sections 4.2 and 4.2 and for practical applications Templ *et al.*

(2014). The values of continuously scaled key variables must be perturbed as well. In this case, micro-aggregation is always a good choice (see Section 4.2). More sophisticated methods such as shuffling (see Section 4.2) often also provide promising results.

5. After modifying categorical and continuous key variables, information loss and disclosure risk measures are estimated. The goal is to release a safe data set with low (individual) risks and high data utility. If the risks are low enough and the data utility is high, the anonymized data set can be released. If not, the entire anonymization process must be reiterated, either with additional perturbations if (some) remaining risks are considered too high or with actions that increase data utility.

Note that other concepts to anonymize micro-data such as the simulation of synthetic micro-data sets exist (see Figure 1). One possibility is to simulate all variables of a micro-data set (i.e., simulate fully synthetic data) by statistical modeling and drawing from predictive distributions. This topic and its application in R is fully covered in Alfons *et al.* (2011) and Templ and Filzmoser (2014) and can be practically carried out using R package **simPopulation** (Alfons and Kraft 2013).

3. Working with sdcMicro

For each method discussed we additionally show its usage via the command line interface of **sdcMicro**. The application of (most) methods is also possible using the graphical user interface of package **sdcMicroGUI**, but this is not shown in this contribution.

3.1. General information about sdcMicro and performance

The first version, version 1.0.0, of the **sdcMicro** package was released in 2007 on the Comprehensive R Archive Network (CRAN, `http://CRAN.R-project.org/package=sdcMicro`) and introduced in Templ (2008). However, this version only included few methods and the package consisted of a collection of some functions that were only applicable to small data sets. The current release, version 4.6.0, is a huge step forward. Almost all methods are implemented in an object-oriented manner (using S4 classes) and have an internal implementation in C++ or based on package **data.table** (Dowle and Short 2013). This allows for efficient high performance computations. The IHSN provided C++ code for many methods which were rewritten from scratch (except `suda2` and `rankSwap`) and integrated into **sdcMicro**.

In Figure 2 the performance improvement with respect to computation time using the current version of **sdcMicro** (4.6.0) as compared to the previous implementation in version 4.0.4 using IHSN C++ code is shown. Note that these calculations are not possible for early versions of **sdcMicro** (Templ 2008) due to missing functionalities. To measure the computation time, a close-to-reality data set on household income and expenditures with 4580 observations on 14 variables was used. This data set is available in **sdcMicro** (`?testdata`). The observations were randomly replicated to enlarge the data set up to 10,000,000 observations. For different numbers of observations, frequency counts and risk (left plot in Figure 2) as well as (heuristic "optimal") local suppression (right plot in Figure 2) were applied independently to each data set. While the IHSN C++ solutions has an exponentional increase in computation time with respect to the number of observations, the new implementation features a "linear growth" for local suppression. The computation time for 10,000,000 observations on four key variables is

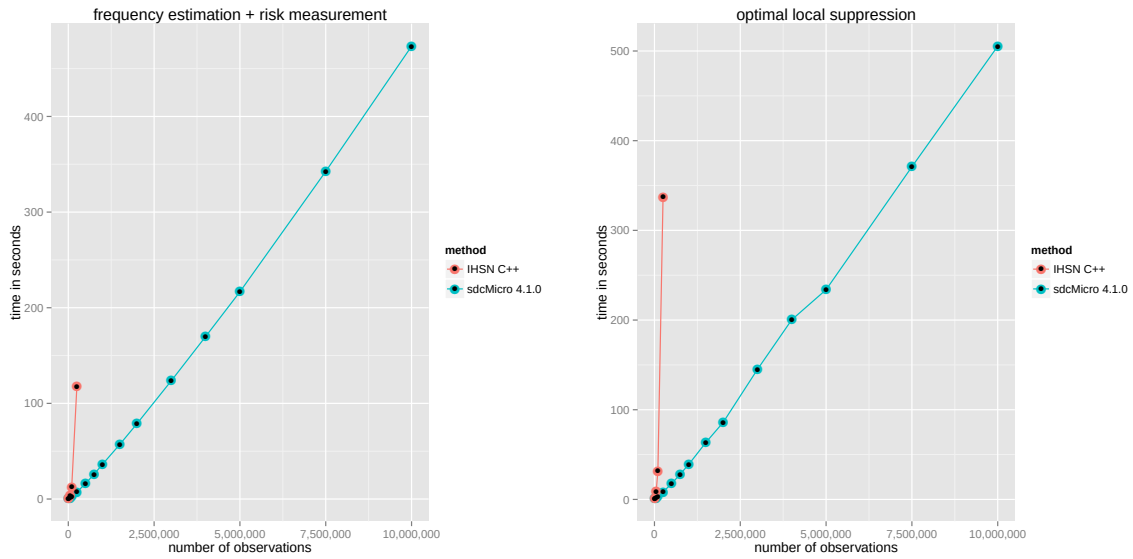

Figure 2: Computation time of IHSN C++ code (equals **sdcMicro** version $< 4.1.0$) and **sdcMicro** (version $\geq 4.1.0$).

approx. 500 seconds with any version of **sdcMicro** $\geq 4.1.0$. On the other hand, with previous versions it is already out of scope to estimate the risk and apply local suppression for data sets with about 500,000 observations.

3.2. S4 class structure

The following list gives an overview about the general aims of **sdcMicro**:

- **sdcMicro** includes the most comprehensive collection of micro-data protection methods;
- the well-defined S4 class implementation provides a user-friendly implementation and makes it easy to exploit its functionalities using **sdcMicroGUI**;
- utility functions extract information from well-defined S4 class objects;
- certain slots are automatically updated after application of a method;
- an undo function allows to return to a previous state of the anonymization process without the need to do additional calculations;
- for performance reasons, the methods are internally implemented either in C++ or by using the **data.table** package (Dowle and Short 2013);
- dynamic reports about results of the anonymization process can be generated.

To define an object of class '`sdcMicroObj`', the function `createSdcObj()` can be used. Parameters for this function are for example categorical and continuous key variables, the vector of sampling weights and optionally stratification and cluster IDs. The following code shows how to generate such an object using a test data set of the International Income Distribution

Data Set (I2D2), which is distributed over the International Household Survey Network and which is also included in **sdcMicro**.

```
R> library("sdcMicro")
R. data("testdata", package = "sdcMicro")
R> sdc <- createSdcObj(testdata,
+    keyVars = c("urbrur", "water", "sex", "age"),
+    numVars = c("expend", "income", "savings"), pramVars = "walls",
+    w = "sampling_weight", hhId = "ori_hid")
```

This shows how to define the categorical and continuous key variables, the index of variables that are selected for PRAM (see Section 4.2), the vector of weights and the household IDs. The following slots of an object of class '`sdcMicroObj`' are pre-filled:

```
R> slotNames(sdc)
```

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

Slot `origData` contains the original data, `keyVars` the index of categorical key variables. `pramVars` contains an index indicating variables that are pramed (see Section 4.2 for details), slot `numVars` specifies indices of continuous key variables and the vector defining sampling weights is contained in slot `weightVar`. A possible index determining the cluster variable (slot `hhId`), the stratification variable (slot `strataVar`) and sensible variables (slot `sensibleVar`) can also be used. In addition, manipulated variables are saved in the slots beginning with `manip`. All risk measures (see Section 4.1) are stored in slots `originalRisk` (for the original unmodified data) and `risk` (for the manipulated data). Slot `utility` collects all information on data utility; further information on pramed variables and local suppressions are stored in slots `pram` and `localSuppression` while additional results (e.g., self-defined utility measures) are saved in slot `additionalResults`. Optionally, in the slot `prev` previous results are saved. Wrapper functions to extract relevant information from the '`sdcMicroObj`' object are available (see Section 3.3). For more details on the structure of the '`sdcMicroObj`' object have a look at the help file (`help("createSdcObj")`).

The show (print) method shows some basic properties of objects of class '`sdcMicroObj`':

```
R> sdc
```

```
Data set with 4580 rows and 14 columns.
Weight variable: sampling_weight

Categorical key variables: urbrur, water, sex, age
```

```
Reported is the
 number | mean size and | size of smallest category

-------------
urbrur .. 2 | 2290 | 646
 -------------
water ... 8 | 572 | 26
 -------------
sex ..... 2 | 2290 | 2284
 -------------
age ..... 88 | 52 | 1

Number of observations violating

 -  2-anonymity:  330
 -  3-anonymity:  674
--------------------------

Percentage of observations violating
 -  2-anonymity:  7.21 %
 -  3-anonymity:  14.72 %


Numerical key variables: expend, income, savings
Disclosure Risk is between:
 [0% ; 100%] (current)
 - Information Loss:
    IL1: 0
  - Difference Eigenvalues: 0 %
```

Methods are applied to the '`sdcMicroObj`' object and all related computations are done automatically. E.g., individual risks are re-estimated whenever a protection method is applied and the related slots of the '`sdcMicroObj`' object are updated. A method for an '`sdcMicroObj`' object can be applied by `method(sdcMicroObj)`, where `method` is a placeholder for a specific method `sdcMicroObj` for an object of class '`sdcMicroObj`'. We note that **sdcMicro** also supports the straightforward application of methods to micro-data. For example, micro-aggregation of three continuous key variables (`expend`, `income` and `savings`) on the data set `testdata` can be achieved with:

```
R> microaggregation(testdata[, c("expend", "income", "savings")])
```

This code is equivalent to `microaggregation(sdc)` where `sdc` is a '`sdcMicroObj`' object and the three variables defined as numeric variables in slot `numVars`. In Table 2 we list the package's current methods, function calls, function description, and slots updated in the '`sdcMicroObj`' object by each method. For example, the application of `localSuppression()` on an '`sdcMicroObj`' object suppresses certain values in the data set and afterwards it updates the slots `risk` and `localSuppression`. In the slot `risk` all relevant list elements are replaced

Function	Aim	Updates the slots
`freqCalc()`	sample and population frequency estimation (used by `measureRisk()`)	–
`suda2()`	frequency calculation on subsets	`@risk$suda2`
`ldiversity()`	l-diversity	`@risk$ldiversity`
`measureRisk()`	individual, household and global risk estimation	`@risk*`
`LLmodGlobalRisk()`	global risk estimation using log-linear models	`@risk$model`
`dRisk()`	disclosure risk for continuous scaled variables	`@risk$numeric`
`dRiskRMD()`	advanced disclosure risk measures for continuous scaled variables	`@risk$numericRMD` (risk on cont. variables)
`dUtility()`	data utility measures	`@utility$*`
`globalRecode()`	anonymization of categorical key variables	`@risk$*`, `@manipKeyVars`, `@prev`
`groupVars()`	anonymization of categorical key variables	`@risk$*`, `@manipKeyVars`, `@prev`
`localSupp()`	univariate local suppression of high risky values	`@risk$*`, `@localSuppression`, `@manipKeyVars`, `@prev`
`localSuppression()`	local suppression to achieve k-anonymity	`@risk$*`, `@localSuppression`, `@manipKeyVars`, `@prev`
`pram()`	swapping values using the post randomization method	`@pram`, `@manipPramVars`, `@prev`
`microaggrGower()`	micro-aggregation on categorical and continuous key variables	`@risk$*`, `@utility$*`, `@manipNumVars`, `@prev`
`topBottomCoding()`	top and bottom coding	`@risk$*`, `@utility$*`, `@manipNumVars`, `@prev`
`addNoise()`	perturbation of continuous variables	`@risk$numeric`), `@utility$*`, `@manipNumVars`, `@prev`
`rankSwapp()`	perturbation of continuous variables	`@risk$numeric`), `@utility$*`, `@manipNumVars`, `@prev`
`mafast()`	perturbation of continuous variables	`@risk$numeric` , `@utility$*`, `@manipNumVars`, `@prev`
`microaggregation()`	perturbation of continuous variables, wrapper for various methods	`@risk$numeric` , `@utility$*`, `@manipNumVars`, `@prev`
`shuffle()`	perturbation of continuous variables	`@risk$numeric`, `@utility$*`, `@manipNumVars`, `@prev`

Table 2: Functions in **sdcMicro** including various SDC methods.

with new estimates and in the slot `localSuppression` the information of suppressed values gets updated. Another example is to apply `microaggregation`.

```
R> sdc <- microaggregation(sdc)
```

Function	Description
`show()`	Print method for objects of class '`sdcMicroObj`'.
`print()`	Print methods showing information on k-anonymity, l-diversity, local suppressions, recoding, disclosure risk, SUDA2 and PRAM
`get.sdcMicroObj()`	Directly returns slots from '`sdcMicroObj`' objects.
`generateStrata()`	Generates a single variable defining strata from multiple variables.
`undolast()`	Reverts the last modification of an object of class '`sdcMicroObj`'.
`extractManipData()`	Extracts manipulated data from '`sdcMicroObj`' objects.
`calcRisks()`	Recomputes the disclosure risk on objects of class '`sdcMicroObj`'.
`varToFactor()` and `varToNumeric()`	Changes a key variable of an object of class '`sdcMicroObj`' from `numeric` to `factor` or from `factor` to `numeric`.

Table 3: Utility functions.

In the above code, method micro-aggregation is applied using default values (to change these, see `help("microaggregation")`) to an object `sdc` of class '`sdcMicroObj`'. Since function `microaggregation()` is only suitable for continuous scaled variables (use `microaggrGower()` for other cases), the categorical variables remain untouched and micro-aggregation is applied on all continuous key variables. Finally, slot `risk$numeric` (disclosure risk for continuous key variables) and slot `ytility$*` (data utility for continuous key variables) and slot `manipNumVars` (the perturbed variables) are updated or filled. In addition, information on the previous state of the anonymization is saved in slot `prev`.

3.3. Utility functions

Available helper functions in **sdcMicro** are listed in Table 3. Functions to extract information from different slots are implemented. Helpful (especially when working with **sdcMicroGUI**), is the `undolast()` function that allows to undo the last step(s) of the anonymization.

The slots of the '`sdcMicroObj`' object can be accessed using function `get.sdcMicroObj()` and the current state of the data (with all anonymizations done so far) can be extracted, see the following code where the data utility, the categorical key variables and the manipulated data are accessed:

```
R> ut <- get.sdcMicroObj(sdc, "utility")
R> cat <- get.sdcMicroObj(sdc, "keyVars")
R> dat <- extractManipData(sdc)
R> str(dat)

'data.frame':  4580 obs. of  14 variables:
 $ urbrur         : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
 $ roof           : int  4 4 4 4 4 4 4 4 4 4 ...
 $ walls          : int  3 3 3 3 2 2 2 2 2 2 ...
 $ water          : Factor w/ 8 levels "1","2","3","4",..: 3 3 3 3 3 3 3 ...
```

```
 $ electcon       : int  1 1 1 1 1 1 1 1 1 ...
 $ relat          : int  1 2 3 3 1 2 3 3 3 ...
 $ sex            : Factor w/ 2 levels "1","2": 1 2 1 1 1 2 2 2 1 2 ...
 $ age            : Factor w/ 88 levels "0","1","2","3",..: 47 42 10 7 53 48
 14 20 10 17 ...
 $ hhcivil        : int  2 2 1 1 2 2 1 1 1 1 ...
 $ expend         : num  89216447 26627836 23178458 16100539 9097752 ...
 $ income         : num  56066667 25933333 67700000 81433333 88200000 ...
 $ savings        : num  198935 478764 5579820 8634905 282855 ...
 $ ori_hid        : int  1 1 1 1 2 2 2 2 2 2 ...
 $ sampling_weight: int  100 100 100 100 100 100 100 100 100 100 ...
```

Print methods are available to show relevant informations. The following code prints results about risk currently stored in object `sdc`. For explanations about risk measures, see Section 4.1.

```
R> print(sdc, "risk")
```

```
---------------------------
0 obs. with higher risk than the main part
Expected no. of re-identifications:
 24.78 [ 0.54 %]
---------------------------
---------------------------
Hierarchical risk
---------------------------
Expected no. of re-identifications:
 117.2 [ 2.56 %]
```

Other print methods report the number and percentage of observations violating 2- and 3-anonymity, the number of local suppressions, information on recodings, the individual and cluster risk, the risk on continuous key variables and information on pramed variables (output is suppressed):

```
R> print(sdc)
R> print(sdc, "ls")
R> print(sdc, type = "recode")
R> print(sdc, type = "risk")
R> print(sdc, type = "numrisk")
R> print(sdc, type = "pram")
```

More information on **sdcMicro** and its facilities can be found in the manual of **sdcMicro**, see Templ *et al.* (2015). The development version is hosted on https://github.com/alexkowa/sdcMicro and includes test batteries to ensure that the package keeps stable when modifying parts of the package. From time to time, a new version is uploaded to CRAN.

4. Methods

In this chapter, a brief review of methods for statistical disclosure control is given. First, different kinds of disclosure risk methods are briefly discussed in Section 4.1 followed by a discussion on anonymization methods in Section 4.2. Concepts of data utility are briefly described in Section 4.3. More in-depth reviews of the procedures mentioned can be found, e.g., in Skinner (2009); Matthews and Harel (2011).

4.1. Measuring the disclosure risk

Measuring risk in micro-data is a key task and is essential to determine if the data set is secure enough to be released. To assess disclosure risks, realistic assumptions about the information data users might have at hand to match against the micro-data set must be made. These assumptions are denoted disclosure risk scenarios. Based on a specific disclosure risk scenario, a set of key variables (i.e., identifying variables) must be defined that are used as input for the risk evaluation procedure.

Based on these key variables, the frequencies of combinations of categories in the key variables are calculated for the sample data and are also estimated on population level. In the following, we focus on the analysis of these frequencies. Some methods only take available frequencies in the sample into account (k-anonymity, l-diversity, SUDA2). More sophisticated methods estimate the risk on estimated/modeled population frequency counts (individual risk approach, log-linear model approach). In any case it is the goal to obtain information on the risk for each individual unit and to summarize it to a global risk measure which evaluates the disclosure risk of the entire data set. Also, different methods can be applied to continuous key variables. These methods focus on the evaluation if perturbed (masked) values are too close to the original values.

Population frequencies and the individual risk approach

Typically, risk evaluation is based on the concept of uniqueness in the sample and/or in the population. The focus is on individual units that possess rare combinations of selected key variables. It is assumed that units having rare combinations of key variables can be more easily identified and thus have a higher risk of re-identification. It is possible to cross-tabulate all identifying variables and view their cast. Keys possessed by only few individuals are considered risky, especially if these observations also show small sampling weights. This means that the expected number of individuals having such keys is also expected to be low in the population.

To assess if a unit is at risk, a threshold approach is typically used. If the re-identification risk of a unit is above a certain value, this unit is said to be at risk. To compute individual risks, the frequency of a given key pattern in the population must be estimated. To illustrate this approach, we consider a random sample of size n drawn from a finite population of size N. Let π_j, $j = 1, \ldots, N$ be the (first order) inclusion probabilities. π_j is the probability that element u_j of a population of the size N is chosen in a sample of size n.

All possible combinations of categories in the key variables (i.e., *keys* or *patterns*) can be calculated by cross-tabulating these variables. Let M be the number of unique keys. For each key, the frequency of observations with the corresponding pattern can be calculated, which results in M frequency counts. Each observation corresponds to one pattern and

the frequency of this pattern. Denote f_i, $i = 1, \ldots, n$ to be the sample frequency count and let F_i be the population frequency count corresponding to the same pattern for the ith observation. If $f_i = 1$ applies, the corresponding observation is unique in the sample given the key variables. If $F_i = 1$, then the observation is unique in the population and the sample. We note, that F_i is usually not known and has to be estimated. A very basic approach to estimate F_i would be to sum up the sampling weights for observations belonging to the same combination of key variables because the weights contain the information on expected units in the population. More sophisticated methods estimate these frequencies based on (log-linear) models, as discussed in a subsequent paragraph.

In the following the methods are applied to the test data set. Before starting, the `undolast()` function is used to undo the last action (micro-aggregation applied in Section 3.2)

```
R> sdc <- undolast(sdc)
```

When creating a '`sdcMicroObj`' object the frequency calculations are done automatically. The general extractor function `get.sdcMicroObj` can be used to extract sample and population frequencies from the current object:

```
R> head(get.sdcMicroObj(sdc, type = "risk")$individual)
```

```
             risk fk   Fk   hier_risk
[1,] 0.0016638935  7  700 0.004330996
[2,] 0.0016638935  7  700 0.004330996
[3,] 0.0005552471 19 1900 0.004330996
[4,] 0.0004543389 23 2300 0.004330996
[5,] 0.0024937656  5  500 0.009682082
[6,] 0.0033222591  4  400 0.009682082
```

Without creating an object of class '`sdcMicroObj`', one can use the function `freqCalc()` for frequency estimation. It basically includes three parameters (for benchmarking issues a fourth parameter is provided) determining the data set, the key variables and the vector of sampling weights (for details, see `?freqCalc`). However, it can be shown that risk measures based on such estimated population frequency counts almost always overestimate small population frequency counts (see, e.g., Templ and Meindl 2010, and Section 4.1).

The concept of k-anonymity

Based on a set of key variables, a desired characteristic of a protected micro-data set is often to achieve k-anonymity (Samarati and Sweeney 1998; Sweeney 2002). This means that for each possible key at least k units in the data set are assigned. This is equal to $f_i \geq k$, $i = 1, \ldots, n$. A typical value is $k = 3$. The default print method of an object of class '`sdcMicroObj`' can be used to see if the current '`sdcMicroObj`' object already features 2- or 3-anonymity:

```
R> print(sdc)
```

```
Number of observations violating
```

	Key 1	Key 2	Sensible variable	f_k	l-diversity
1	1	1	50	3	2
2	1	1	50	3	2
3	1	1	42	3	2
4	1	2	42	1	1
5	2	2	62	2	1
6	2	2	62	2	1

Table 4: k-anonymity and l-diversity on a toy data set.

```
 -  2-anonymity:  330
 -  3-anonymity:  674
--------------------------

Percentage of observations violating
 -  2-anonymity:  7.21 %
 -  3-anonymity:  14.72 %
```

k-anonymity is typically achieved by recoding categorical key variables into fewer categories and by suppressing specific values of key variables for some units (see Sections 4.2 and 4.2).

l-diversity

An extension of k-anonymity is l-diversity (Machanavajjhala, Kifer, Gehrke, and Venkitasubramaniam 2007). Consider a group of observations with the same pattern in the key variables and let the group fulfill k-anonymity. A data intruder can therefore by definition not identify an individual within this group. If, however, all observations have the same entries in an additional sensitive variable (e.g., *cancer* in the variable *medical diagnosis*), an attack will be successful if the attacker can identify at least one individual of the group, as the attacker knows that this individual has cancer with certainty. The distribution of the target-sensitive variable is referred to as l-diversity.

Table 4 considers a small example data set that shows the calculation of l-diversity and points out the slight difference of this measure as compared to k-anonymity. The first two columns of Table 4 contain the categorical key variables. The third column of the data defines a variable containing sensitive information. Sample frequency counts f_i appear in the fourth column. They equal 3 for the first three observations; the fourth observation is unique and frequency counts f_i are 2 for the last two observations. Only the fourth observation violates 2-anonymity. Looking closer at the first three observations, we see that only two different values are present in the sensitive variable. Thus the l-(distinct) diversity is just 2. For the last two observations, 2-anonymity is achieved, but the intruder still knows the exact information of the sensitive variable. For these observations, the l-diversity measure is 1, indicating that sensitive information can be disclosed since the value of the sensitive variable equals 62 for both of these observations.

Differences in values of the sensitive variable can be measured differently. We present here the distinct diversity that counts how many different values exist within a pattern/key. The l-diversity measure is automatically measured in **sdcMicro** for (and stored in) objects of class '`sdcMicroObj`' as soon as a sensible variable is specified (using `createSdcObj`). Note that

the measure can be calculated at any time using `ldiversity(sdc)` with optional function parameters to select another sensible variable, see `?ldiversity`. However, it can also be applied to data frames, where key variables (argument `keyVars`) and the sensitive variables (argument `ldiv_index`) must be specified as shown below:

```
R> res1 <- ldiversity(testdata, keyVars = c("urbrur", "water", "sex", "age"),
+    ldiv_index = "income")
R> print(res1)

--------------------------
L-Diversity Measures
--------------------------
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    4.00    8.00   10.85   17.00   35.00
```

Additional methods such as entropy, recursive and multi-recursive measures of l-diversity are also implemented. For more information, see the help files of **sdcMicro**.

Sample frequencies on subsets: SUDA2

SUDA (i.e., special uniques detection algorithm) estimates disclosure risks for each unit. SUDA2 (Manning, Haglin, and Keane 2008) is a recursive algorithm to find minimal sample uniques. The algorithm generates all possible variable subsets of selected (categorical) key variables and scans for unique patterns within subsets of these variables. The risk of an observation finally depends on two aspects:

(a) The lower the number of variables needed to receive uniqueness, the higher the risk (and the higher the SUDA score) of the corresponding observation.

(b) The larger the number of minimal sample uniqueness contained within an observation, the higher the risk of this observation.

Item (a) is calculated for each observation i by $l_i = \prod_{k=MSUmin_i}^{m-1}(m-k)$, $i = 1, \ldots, n$. In this formula, m corresponds to the *depth*, which is the maximum size of variable subsets of the key variables, $MSUmin_i$ is the number of minimal sample uniques (MSU) of observation i and n is the number of observations in the data set. Since each observation is treated independently, the specific values l_i belonging to a specific pattern are summed up. This results in a common SUDA score for all observations having this pattern; this summation is the contribution of Item (b).

The final SUDA score is calculated by normalizing these SUDA scores by dividing them by $p!$, with p being the number of key variables. To receive the so-called data intrusion simulation (DIS) score, loosely speaking, an iterative algorithm based on sampling of the data and matching of subsets of the sampled data with the original data is applied. This algorithm calculates the probabilities of correct matches given unique matches. It is, however, out of scope to precisely describe this algorithm here, see Elliot (2000) for details. The DIS SUDA score is calculated from the SUDA and DIS scores and is available in **sdcMicro** as `disScore`. Note that this method does not consider population frequencies in general, but considers sample frequencies on subsets. The DIS SUDA scores approximate uniqueness by

simulation based on the sample information population but – to our knowledge – generally do not consider sampling weights. Thus, biased estimates may result.

SUDA2 is implemented in **sdcMicro** as function `suda2()` based on C++ code from the IHSN. Additional output, such as the contribution percentages of each variable to the score, are also available as an output of this function. The contribution to the SUDA score is calculated by assessing how often a category of a key variable contributes to the score. After an object of class '`sdcMicroObj`' has been created, no information about SUDA (DIS) scores are stored. However, after applying SUDA on such an object, they are available, see the following code where also the print method is used:

```
R> sdc <- suda2(sdc)
R> get.sdcMicroObj(sdc, type = "risk")$suda

 Dis suda scores table:
- - - - - - - - - - -
  thresholds number
1        > 0   4330
2      > 0.1    250
3      > 0.2      0
4      > 0.3      0
5      > 0.4      0
6      > 0.5      0
7      > 0.6      0
8      > 0.7      0
 - - - - - - - - - - -
```

The individual SUDA scores and DIS scores are accessible in slots `risk$suda2$score` and `risk$suda2$disScore` as well as the contribution of each key variable to the SUDA score for each observation in slot `risk$suda2$contributionPercent`.

The individual and cluster risk approach

An approach that considers sampling weights is the individual risk approach. It uses so-called super-population models. In such models population frequency counts are modeled given a certain distribution. The estimation procedure of sample counts given the population counts can be modeled for example by assuming a negative binomial distribution (see Rinott and Shlomo 2006) and is implemented in **sdcMicro** in function `measure_risk()` (for details, see Templ and Meindl 2010). This function can be explicitly used for data frames. For objects of class '`sdcMicroObj`' this function is internally applied when creating the object and each time when categorical key variables are modified to update the `risk` slot in the object.

Micro-data sets often contain hierarchical cluster structures, e.g., individuals that are clustered in households. The risk of re-identifying an individual within a household may also affect the probability of disclosure of other members in the same household. Thus, the household or cluster structure of the data must be taken into account when calculating risks. It is commonly assumed that the risk of re-identification of a household is the risk that at least one member of the household can be disclosed. Thus this probability can be simply estimated

from individual risks as 1 minus the probability that no member of the household can be identified.

This is also the implementation strategy in **sdcMicro**. The individual and cluster/hierarchical risks are stored together with sample (f_k) and population counts (F_k) in slot `risk$individual` and can be extracted by function `get.sdcMicroObj` as shown below:

```
R> head(get.sdcMicroObj(sdc, "risk")$individual)
```

```
             risk fk   Fk   hier_risk
[1,] 0.0016638935  7  700 0.004330996
[2,] 0.0016638935  7  700 0.004330996
[3,] 0.0005552471 19 1900 0.004330996
[4,] 0.0004543389 23 2300 0.004330996
[5,] 0.0024937656  5  500 0.009682082
[6,] 0.0033222591  4  400 0.009682082
```

Measuring the global risk

In the previous section, the theory of individual risks and the extension of this approach to clusters such as households were discussed. In many applications it is preferred to estimate a measure of global risk. Any global risk measure will result in a single number that can be used to assess the risk of an entire micro-data set.

Measuring the global risk using individual risks

Two approaches can be used to determine the global risk for a data set using individual risks:

Benchmark: This approach counts the number of observations that can be considered risky and also have higher risk as the main part of the data. For example, we consider units with individual risks being ≥ 0.1 and twice as large as the median of all individual risks $+ 2 \cdot$ median absolute deviation (MAD) of all unit risks.

Global risk: The sum of the individual risks in the data set gives the expected number of re-identifications (see Hundepool *et al.* 2008).

The benchmark approach indicates whether the distribution of individual risk occurrences contains extreme values. This relative measure depends on the distribution of individual risks. It is not valid to conclude that units with higher risk than this benchmark have high risk. The measure evaluates if some unit risks behave differently compared to most of the other individual risks. The global risk approach is based on an absolute measure of risk. Beneath is the print output of the corresponding function from **sdcMicro** showing both measures:

```
R> print(sdc, "risk")
```

```
---------------------------
0 obs. with higher risk than the main part
Expected no. of re-identifications:
```

```
 24.78 [ 0.54 %]
--------------------------
--------------------------
Hierarchical risk
--------------------------
Expected no. of re-identifications:
 117.2 [ 2.56 %]
```

If a cluster (e.g., households) has been defined, a global risk measurement taking into account this hierarchical structure is also reported.

Measuring the global risk using log-linear models

Sample frequencies, considered for each pattern m out of M patterns, f_m, $m = 1, \ldots, M$, can be modeled using a Poisson distribution. In this case, a global risk measure can be defined as:

$$\tau_1 = \sum_{m=1}^{M} \exp\left(-\frac{\mu_m(1-\pi_m)}{\pi_m}\right), \quad \text{with } \mu_m = \pi_m \lambda_m, \tag{1}$$

where π_m are inclusion probabilities corresponding to a pattern m and λ_m correspond to means of Poisson random variables. This risk measure aims at calculating the number of sample uniques that are also population uniques taking into account a probabilistic Poisson model. For details and derivation of the formula we refer to Skinner and Holmes (1998). For simplicity, all (first order) inclusion probabilities are assumed to be equal, $\pi_m = \pi$, $m = 1, \ldots, M$. τ_1 can be estimated by log-linear models that include both the primary effects and possible interactions. This model is defined as:

$$\log(\pi_m \lambda_m) = \log(\mu_m) = \mathbf{x}_m \beta.$$

To estimate the μ_ms, the regression coefficients β have to be estimated using, e.g., iterative proportional fitting. The quality of this risk measurement approach depends on the number of different keys that result from cross-tabulating all key variables. If the cross-tabulated key variables are sparse in terms of how many observations have the same patterns, predicted values might be of low quality. It must also be considered that if the model for prediction is weak, the quality of the prediction of the frequency counts is also weak. Thus, the risk measurement with log-linear models may lead to acceptable estimates of global risk only if not too many key variables are selected and if good predictors are available in the data set.

In **sdcMicro**, global risk measurement using log-linear models can be achieved with function `LLmodGlobalRisk()`. This function is experimental and needs further testing. Thus, it should be used by expert users only. In the following function call an object of class '`sdcMicroObj`' is used as input for function `LLmodGlobalRisk()`. In addition, a model/predictors (using function argument `form`) can be specified to estimate frequencies of the categorical key variables. The reported global risk corresponds to Equation 1.

```
R> sdc <- LLmodGlobalRisk(sdc, form = ~ urbrur + water + sex + age)
R> get.sdcMicroObj(sdc, "risk")$model$gr1

[1] 0.03764751
```

Note that `get.sdcMicroObj(sdc, "risk")$model$gr2` gives access to a more sophisticated risk measure, τ_2, which may be interpreted as the expected number of correct matches for sample uniques. We refer to Skinner and Holmes (1998) and Rinott and Shlomo (2006) for more details.

Measuring risk for continuous key variables

The concepts of uniqueness and k-anonymity cannot be directly applied to continuous key variables because many units in the data set would be identified as unique and the naive approach will fail. The next sections present methods how to measure risks in this case.

If detailed information about a value of a numerical variable is available, attackers may be able to identify and eventually obtain further information about an individual. Thus, an intruder may identify statistical units by applying, for example, linking or matching algorithms. The anonymization of continuous key variables should avoid the possibility of successfully merging the underlying micro-data with other external data sources.

We assume that an intruder has information about a statistical unit included in the micro-data and this information overlaps on some variables with the information in the data. In simpler terms, we assume that the information the intruder possesses can be merged with micro-data that should be secured. In addition, we also assume that the intruder is sure that the link to the data is correct, except for micro-aggregated data (see Section 4.2). In this case, an attacker cannot be sure if the link is valid because at least k observations have the same value for each continuous variable.

The underlying idea of distance-based record linkage methods is to find the nearest neighbors between observations of two different data sets. Mateo-Sanz, Sebe, and Domingo-Ferrer (2004) introduced distance-based record linkage and interval disclosure. In the first approach, they look for the nearest neighbor from each observation of the masked data value to the original data points. Then they mark those units for which the nearest neighbor is the corresponding original value. In the second approach - which is also the default risk of **sdcMicro** – they check if the original value falls within an interval centered on the masked value. In this case, the intervals are calculated based on the standard deviation of the variable under consideration.

Distance-based risks for continuous key variables are automatically estimated for objects of class '`sdcMicroObj`' using function `dRisk()`. Current risks can be printed with:

```
R> print(sdc, "numrisk")

Disclosure Risk is between:
 [0% ; 100%] (current)
 - Information Loss:
    IL1: 0
  - Difference Eigenvalues: 0 %
```

This reports the percentage of observations falling within an interval centered on its masked value where the upper bound corresponds to a worst case scenario where an intruder is sure that each nearest neighbor is indeed the true link. For a more detailed discussion we refer to Mateo-Sanz *et al.* (2004). Since no anonymization has been applied to the continuous key variables, the disclosure risk can be high (up to 100%) and the information loss (discussed later in Section 4.3), is 0.

Almost all data sets used in official statistics contain units whose values in at least one variable are quite different from the general observations and are asymmetrically distributed. Examples of such outliers might be enterprises with very high values for turnover or persons with extremely high income. Also, multivariate outliers exist (see Templ and Meindl 2008a). Unfortunately, intruders may want to disclose a large enterprise or an enterprise with specific characteristics. Since big enterprises are often sampled with certainty, intruders can often be very confident that the enterprise they want to disclose has been sampled. In contrast, an intruder may not be as interested to disclose statistical units that exhibit the same behavior as most other observations. For these reasons it is good practice to define measures of disclosure risk that take the outlyingness of a unit into account, for details, see Templ and Meindl (2008a). The key idea is to assume that outliers should be much more perturbed than non-outliers because these units are easier to re-identify even when the distance from the masked observation to its original observation is relatively large. One key aspect is therefore outlier detection using robust Mahalanobis distances (see, e.g., Maronna, Martin, and Yohai 2006). In this approach it is also considered that micro-aggregation provides anonymization differently, see Templ and Meindl (2008a) for details.

The methods just discussed are also available in **sdcMicro** in function `dRiskRMD()`. While `dRisk()` is automatically applied to objects of class 'sdcMicroObj', `dRiskRMD()` has to be called once to fill the corresponding slot:

```
R> sdc <- dRiskRMD(sdc)
```

The reason of not automatically estimating this risk is that robust estimations are computationally expensive for large data sets since the estimation is based on a robust fit of a covariance matrix using the minimum covariance estimator (mcd). Whenever this risk measure is estimated, the results are saved in slot `risk$numericRMD` or can alternatively extracted using `get.sdcMicroObj(sdc, "risk")$numericRMD`.

4.2. Anonymization methods

Generally, two kinds of anonymization methods can be distinguished: deterministic and probabilistic. For categorical variables, recoding and local suppression are deterministic procedures while swapping and PRAM (Gouweleeuw *et al.* 1998) are based on randomness and considered probabilistic methods. For continuous variables, micro-aggregation is a deterministic method while adding correlated noise (Brand 2002) and shuffling (Muralidhar, Parsa, and Sarathy 1999) are probabilistic procedures. Whenever probabilistic methods are applied, the random seed of the underlying pseudo-random number generator should be fixed to ensure reproducibility of the results.

Recoding

Global recoding is a non-perturbative method that can be applied to categorical and continuous key variables. The basic idea of recoding a categorical variable is to combine several categories into a new, less informative category. If the method is applied to a continuous variable, it means to discretize the variable. The goal in both cases is to reduce the total number of possible outcomes of a variable. Typically, recoding is applied to categorical variables to reduce the number of categories with only few observations.

The categorical key variable `age` selected before has a lot of categories and some categories are sparse. Recoding `age` into `age classes` will cause that the number of observations in the keys increase and less observations are violating the 2- and 3-anonymity assumption. This can be easily seen when comparing the result of `print(sdc)` to the previous `print(sdc)` call in Section 4.1. The actual values and percentage of observations violating 2 and 3-anonymity as well as the original values and percentages are reported so that the user gets information about the effect of the applied anonymization method.

```
R> sdc <- globalRecode(sdc, column = "age",
+    breaks = c(1, 9, 19, 29, 39, 49, 59, 69, 100), labels = 1:8)
R> print(sdc)

Number of observations violating

 -  2-anonymity:  10 (orig:  330 )
 -  3-anonymity:  24 (orig:  674 )
--------------------------

Percentage of observations violating
 -  2-anonymity:  0.22 %  (orig:  7.21 % )
 -  3-anonymity:  0.52 %  (orig:  14.72 % )
```

A special case of global recoding is top and bottom coding, which can be applied to ordinal or continuous variables. The idea for this approach is that all values above (i.e., top coding) and/or below (i.e., bottom coding) a pre-specified threshold value are combined into a new category. Function `globalRecode()` can be applied in **sdcMicro** to perform both global recoding and top/bottom coding. A help file with examples is accessible using `?globalRecode`. We note that **sdcMicroGUI** offers a more user-friendly way of applying global recoding.

Local suppression

Local suppression is a non-perturbative method. It is typically applied to categorical variables to suppress certain values in at least one variable. Input variables are usually part of the categorical key variables that are also used to calculate individual risks as described in Section 4.1. Individual values are suppressed in a way that the set of variables with a specific pattern are increased. Local suppression is often used to achieve k-anonymity, as described in Section 4.1. Using function `localSupp()` of **sdcMicro**, it is possible to suppress values of a key variable for all units with individual risks above a pre-defined threshold, given a disclosure scenario. This procedure requires user intervention by setting the threshold. To automatically suppress a minimum amount of values in the key variables to achieve k-anonymity, one can use function `localSuppression()`.

```
R> sdc <- localSuppression(sdc)
R> print(sdc, "risk")

--------------------------
0  (orig: 0 ) obs. with higher risk than the main part
```

```
Expected no. of re-identifications:
 1.27 [ 0.03 %]  (orig: 24.78 [ 0.54 %])
--------------------------
--------------------------
Hierarchical risk
--------------------------
Expected no. of re-identifications:
 6.53 [ 0.14 %]  (orig: 117.2 [ 2.56 %])

R> print(sdc, "ls")

urbrur .. 0 [ 0 %]

water ... 9 [ 0.197 %]

sex ..... 0 [ 0 %]

age ..... 188 [ 4.105 %]
```

In this implementation, a heuristic algorithm is called to suppress as few values as possible. It is possible to specify a desired ordering of key variables (have a look at the function arguments in `localSuppression()`) in terms of importance, which the algorithm takes into account. It is even possible to specify key variables that are considered of such importance that almost no values for these variables are suppressed.

By specifying the importance of variables as a parameter in `localSuppression()`, for key variables with high importance, suppression will only take place if no other choices are possible which is for example useful if a scientific use file with specific requirements must be produced. Still, it is possible to achieve k-anonymity for selected key variables in almost all cases.

Post-randomization

PRAM (Gouweleeuw *et al.* 1998) is a perturbation, probabilistic method that can be applied to categorical variables. The idea is to recode values of a categorical variable in the original micro-data file into other categories by taking into account pre-defined transition probabilities. The process is modeled using a known transition matrix. For each category of a categorical variable, the matrix lists probabilities to change into other possible outcomes.

As an example, consider a variable with only 3 categories: A1, A2 and A3. The transition of a value from category A1 to category A1 is, for example, fixed with probability $p_1 = 0.85$, which means that only with probability $p_1 = 0.15$ can a value of A1 be changed to either A2 or A3. The probability of a change from category A1 to A2 might be fixed with probability $p_2 = 0.1$ and changes from A1 to A3 with $p_3 = 0.05$. Probabilities to change values from class A2 to other classes and also from A3, to others must be specified beforehand. All transition probabilities must be stored in a matrix that is the main input to function `pram()` in **sdcMicro**. This following example uses the default parameters of `pram()` rather than a custom transition matrix. The default parameters are 0.8 for the minimum diagonal entries for the generated transition matrix and 0.5 for the amount of perturbation for the invariant PRAM method. We can observe from the following output that exactly one value changed

the category. One observation having A3 in the original data has value A1 in the masked data.

```
R> set.seed(1234)
R> A <- as.factor(rep(c("A1", "A2", "A3"), each = 5))
R> A

 [1] A1 A1 A1 A1 A1 A2 A2 A2 A2 A2 A3 A3 A3 A3 A3
Levels: A1 A2 A3
```

We apply `pram()` on vector `A` and print the result:

```
R> Apramed <- pram(A)
R> Apramed

Number of changed observations:
- - - - - - - - - - - -
x != x_pram : 1 (6.67%)
```

The summary provides more detail. It shows a table of original frequencies and the corresponding table after applying PRAM. All transitions that took place are also listed:

```
R> summary(Apramed)

 ----------------------
 original frequencies:

A1 A2 A3
 5  5  5

 ----------------------
 frequencies after perturbation:

A1 A2 A3
 6  5  4

 ----------------------
 transitions:
  transition Frequency
1          1         5
2          2         5
3          3         5
```

PRAM is applied to each observation independently and randomly. This means that different solutions are obtained for every run of PRAM if no seed is specified for the pseudo-random number generator. A main advantage of the PRAM procedure is the flexibility of the method. Since the transition matrix can be specified freely as a function parameter, all desired effects

can be modeled. For example, it is possible to prohibit changes from one category to another by setting the corresponding probability in the transition matrix to 0.

In **sdcMicro**, `pram()` allows PRAM to be performed. The corresponding help file can be accessed by typing `?pram` into an R console. When using the function it is possible to apply PRAM to sub-groups of the micro-data set independently. In this case, the user needs to select the stratification variable defining the sub-groups. If the specification of this variable is omitted, the PRAM procedure is applied to all observations in the data set.

```
R> sdc <- pram(sdc)
R> print(sdc, "pram")

Number of changed observations:
- - - - - - - - - - -
walls != walls_pram : 439 (9.59%)

Number of changed observations:
- - - - - - - - - - -
  variable nrChanges percChanges
1    walls       439        9.59
```

Micro-aggregation

Micro-aggregation is a perturbative method that is typically applied to continuous variables. The idea is that records are partitioned into groups; within each group, the values of each variable are aggregated. Typically, the arithmetic mean is used to aggregate the values, but other methods to calculate the mean are also possible (e.g., the median). Individual values of the records for each variable are replaced by the group aggregation value. Depending on the method chosen in function `microaggregation()`, additional parameters can be specified. For example, it is possible to specify the number of observations that should be aggregated as well as the statistic used to calculate the aggregation. This statistic defaults to be the arithmetic mean. It is also possible to perform micro-aggregation independently to pre-defined clusters, see the following code. The new values of the disclosure risk are printed again to see the effect of micro-aggregation.

```
R> sdc <- microaggregation(sdc, aggr = 4, strata_variables = "age",
+    method = "mdav")
R> print(sdc, "numrisk")

Disclosure Risk is between:
 [0% ; 2.77%] (current)

               (orig: ~100%)
- Information Loss:
    IL1: 0.42
  - Difference Eigenvalues: 1.5 %

 (orig: Information Loss: 0)
```

We can observe that the disclosure risk decreased considerably ($\sim$ only 2.77% of the original values do not fall within intervals calculated around the perturbed values, compare Section 4.1 on distance-based risk estimation). We can also see that the information loss criteria increased slightly. All of the previous settings (and many more) can be applied in **sdcMicro**. The corresponding help file can be viewed with command `?microaggregation`.

Not very commonly used is the micro-aggregation of categorical and continuous variables. This can be achieved with function `microaggrGower` that uses the Gower distance (Gower 1971) to measure the similarity between observations. For details have a look at the help function `?microaggrGower` in R.

Adding noise

Adding noise is a perturbative protection method for micro-data which is typically applied to continuous variables. This approach can protect data against exact matching with external files if, for example, information on specific variables is available from registers. While this approach sounds simple in principle, many different algorithms can be used to overlay data with stochastic noise. It is possible to add uncorrelated random noise. In this case, the noise is usually normally distributed and the variance of the noise term is proportional to the variance of the original data vector. Adding uncorrelated noise preserves means but variances and correlation coefficients between variables cannot be preserved. The later statistical property is respected, however, if correlated noise method(s) are applied.

For the correlated noise method (Brand 2002), the noise term is derived from a distribution having a covariance matrix that is proportional to the co-variance matrix of the original micro-data. In the case of correlated noise addition, correlation coefficients are preserved and the covariance matrix can at least be consistently estimated from the perturbed data. However, the data structure may differ a great deal if the assumption of normality is violated. Since this is virtually always the case when working with real-world data sets, a robust version of the correlated noise method is included in **sdcMicro**. This method allows departures from model assumptions and is described in detail in Templ and Meindl (2008b).

We now want to apply a method for adding correlated noise based on non-perturbed data after we undo micro-aggregation:

```
R> sdc <- undolast(sdc)
R> sdc <- addNoise(sdc, method = "correlated2")
R> print(sdc, "numrisk")

Disclosure Risk is between:
 [0% ; 32.84%] (current)

                (orig: ~100%)
- Information Loss:
    IL1: 0.11
  - Difference Eigenvalues: 0.88 %

 (orig: Information Loss: 0)
```

We see that the data utility measure is comparable with the micro-aggregation on strata

in the previous code chunk (see the `print(sdc, "numrisk")` output above) but the risk is higher when adding correlated noise.

In **sdcMicro**, many algorithms are implemented that can be used to add noise to continuous variables. For example, it is possible to add noise only to outlying observations. In this case it is assumed that such observations possess higher risks than non-outlying observations. Other methods ensure that the amount of noise added takes into account the underlying sample size and sampling weights. Noise can be added to variables using function `addNoise()` and the corresponding help file can be accessed with `?addNoise`.

Shuffling

Various masking techniques based on linear models have been developed in the literature, such as multiple imputation (Rubin 1993), general additive data perturbation (Muralidhar *et al.* 1999) and the information preserving statistical obfuscation synthetic data generators (Burridge 2003). These methods are capable of maintaining linear relationships between variables but fail to maintain marginal distributions or non-linear relationships between variables.

Shuffling (Muralidhar and Sarathy 2006) simulates a synthetic value of the continuous key variables conditioned on independent, non-confidential variables. After the simulation of the new values for the continuous key variables, reverse mapping (shuffling) is applied. This means that ranked values of the simulated values are replaced by the ranked values of the original data (columnwise). In the implementation of **sdcMicro**, a model of almost any form and complexity can be specified (see `?shuffling` for details) and different methods are available.

In the following code we do not use default values because we want to show how to specify the form of the model. We first restore the previous results and remove the effect of adding noise using `undolast()`.

```
R> sdc <- undolast(sdc)
R> form <- formula(expend + income + savings ~ age + urbrur + water +
+    electcon + age + sex, data = testdata)
R> sdc <- shuffle(sdc, form)
R> print(sdc, "numrisk")
```

```
Disclosure Risk is between:
 [0% ; 0.22%] (current)

                (orig: ~100%)
- Information Loss:
    IL1: 1.64
  - Difference Eigenvalues: 3.51 %

 (orig: Information Loss: 0)
```

Finding a good model is essential for providing good results. If we added for example variable `walls` to the model, the results would no longer be of high quality.

4.3. Measuring data utility

Measuring data utility of the micro-data set after disclosure limitation methods have been applied is encouraged to assess the impact of these methods.

Generally applicable methods

Anonymized data should have the same structure as the original data and should allow any analysis with high precision. To evaluate the precision, various classical estimates such as means and covariances can be used. With function `dUtility()` it is possible to calculate different measures based on classical or robust distances for continuous scaled variables. Estimates are computed for both the original and perturbed data and then compared. We now discuss two important information loss measures:

- **IL1** is a measures introduced by Mateo-Sanz *et al.* (2004). This measure is given as

$$IL1 = \frac{1}{p}\sum_{j=1}^{p}\sum_{i=1}^{n}\frac{|x_{ij} - x_{ij}^{'}|}{\sqrt{2}S_j}$$

  with x_{ij} and $x_{ij}^{'}$ the values of the original data set $\mathbf{X}$ and perturbed data set $\mathbf{X}^{'}$ of sizes $n \times p$, and S_j the standard deviation of variable $\mathbf{X}_j$. It can be interpreted as scaled distances between original and perturbed values for all p continuous key variables;

- **eigen** is a measure calculating relative absolute differences between eigenvalues of the covariances from standardized continuous key variables of the original and perturbed variables. Eigenvalues can be estimated from a robust or classical version of the covariance matrix.

Note that these measures are automatically estimated in **sdcMicro** when an object of class '`sdcMicroObj`' is generated or whenever continuous key variables are modified in such an object. Thus, no user input is needed. The data utility measures are shown when printing the risk (see previous chunks) but they can also be explicitly extracted:

```
R> get.sdcMicroObj(sdc, "utility")

$il1
[1] 1.635679

$eigen
[1] 0.03509664
```

A note on other measures of data utility

In practice it is not possible to create an anonymized file with the same structure as the original file. An important goal, however, should always be that the difference in results of the most important statistics based on anonymized and original data should be very small or even zero. Thus, the goal is to measure data utility based on benchmarking indicators (Ichim

and Franconi 2010; Templ 2011b), which is in general a better approach to assess data quality than applying general tools.

The first step in quality assessment is to evaluate what users of the underlying data are analyzing and then try to determine the most important estimates or *benchmarking indicators* (see, e.g., Templ 2011a,b). Special emphasis should be put on benchmarking indicators that take into account the most important variables of the micro-data set. Indicators that refer to the most sensitive variables within the micro-data should also be calculated. The general procedure is quite simple and can be described in the following steps:

- selection of a set of benchmarking indicators;
- choice of a set of criteria as to how to compare the indicators;
- calculation of all benchmarking indicators of the original micro data;
- calculation of the benchmarking indicators on the protected micro-data set;
- comparison of statistical properties such as point estimates, variances or overlaps in confidence intervals for each benchmarking indicator;
- assessment as to whether the data utility of the protected micro data set is good enough to be used by researchers.

If the quality assessment in the last step of the sketched algorithm is positive, the anonymized micro-data set is ready to be published. If the deviations of the main indicators calculated on the original and the protected data are too large, the anonymization procedure should be restarted and modified. It is possible to either change some parameters of the applied procedures or start from scratch and completely change the anonymization strategy.

Usually the evaluation is focused on the properties of numeric variables, given unmodified and modified micro-data. It is of course also possible to review the impact of local suppression or recoding that has been conducted to reduce individual re-identification risks. Another possibility to evaluate data utility of numerical variables is to define a model that is fitted on the original micro-data. The idea is to predict important, sensitive variables using this model both for the original and protected micro-data set as a first step. In a second step, statistical properties of the model results, such as the differences in point estimates or variances, are compared for the predictions, given original and modified micro-data. Finally, the quality of the results is assessed. If the deviations are small enough, one may go on to publish the protected micro-data set. Otherwise, adjustments must be made in the protection procedure.

In addition, it is interesting to evaluate the set of benchmarking indicators not only for the entire data set but also independently for subsets of the data. In this case, the micro-data are partitioned into a set of h groups. The evaluation of benchmarking indicators is then performed for each of the groups and the results are evaluated by reviewing differences between indicators for original and modified data in each group.

The slot `additionalResults` can be used to store additional results such as self-defined indicators in an object of class '`sdcMicroObj`'. This is sometimes useful because all results corresponding to an anonymization process are stored in one single object. The following line of code estimates the gender pay gap using R package **laeken** (Alfons and Templ 2013), and stores the result within the object `sdc`.

```
R> library("laeken")
R> sdc@additionalResults$gpg <- gpg(inc = "income", method = "mean",
+    gender = "sex", weights = "sampling_weight", breakdown = "water",
+    data = extractManipData(sdc))$valueByStratum$value
```

This result can then be simply compared for each category of `water` with the result of the original data to get an indicator of data utility expressed as relative absolute errors (in percentages):

```
R> testdata$sex <- as.factor(testdata$sex)
R> res <- gpg(inc = "income", method = "mean", gender = "sex",
+    weights = "sampling_weight", breakdown = "water",
+    data = testdata)$valueByStratum$value
R> options(scipen = 999)
R> 100 * abs((res - sdc@additionalResults$gpg)/res)

[1]   30.28450  403.90591   51.22580   21.85279 1867.59048   13.71210
[7]  131.11358  280.25468
```

We also can undo the last step of the anonymization procedure, apply a different disclosure limitation technique and re-calculate the test statistics. In this case we note that the impact of micro-aggregation on the benchmarking indicator is smaller than the impact of shuffling.

```
R> sdc <- undolast(sdc)
R> sdc <- microaggregation(sdc)
R> sdc@additionalResults$gpg <- gpg(inc = "income", method = "mean",
+    gender = "sex", weights = "sampling_weight", breakdown = "water",
+    data = extractManipData(sdc))$valueByStratum$value
R> 100 * abs((res - sdc@additionalResults$gpg) / res)

[1]  2.694697  8.391268  1.342160  2.777190  9.043971 14.527815 20.265172
[8] 80.290284
```

5. Reporting facilities

Using the function `report()` it is possible to generate reports about the results of the anonymization process. The reports are internally generated using packages **brew** (Horner 2011) and **knitr** (Yihui 2013). Two different types, internal (setting function argument `internal = TRUE`) and external (setting function argument `internal = FALSE`) reports, can be produced and exported as HTML, PDF or plain text files. `args(report)` shows the possible function arguments and `?report` gives access to the help file. It is for example possible to select the output format and the output directory of the resulting report.

Internal reports include information about selected key variables, performed actions, disclosure risk and data utility and session information on the package versions used. This detailed report is suitable and useful for the organization that holds the data for internal use and documentation of the anonymization procedure.

External reports contain less information than internal reports. For example, all information about disclosure risks and information loss is suppressed. This report is suitable for external users of the anonymized micro-data set.

All the information that is included in the report always depends on the anonymization process that has been applied and reflects the current values in a given object of class '`sdcMicroObj`'. For example, if PRAM was not applied, no specific summary for variables subjected to PRAM is provided because this information is not available. However, if PRAM was applied, the entire disclosure risk summary is presented differently because the usual risk measures are not valid if categorical key variables have been modified using this procedure. The output of `report(sdc, format = "LATEX")` and `report(sdc, internal = TRUE)` can be obtained using the supplementary material.

6. Conclusion

In this paper, the R package **sdcMicro** was described in detail. This package implements popular statistical disclosure methods for risk estimation such as the SUDA2 algorithm, the individual risk approach or risk measurement using log-linear models. In addition, perturbation methods such as global recoding, local suppression, post-randomization, micro-aggregation, adding correlated noise, shuffling and various other methods are integrated.

With package **sdcMicro**, statistical disclosure control methods can be applied in an exploratory, interactive and user-friendly manner. All results are saved in a structured S4 class object that contains all relevant information. Many useful methods are defined for this class and can be applied to objects of such a class. All relevant slots of an object of class '`sdcMicroObj`' are updated automatically whenever an anonymization method is applied. Print and summary methods are defined and ready to be used to summarize the status of disclosure risk and data utility. Additionally, the whole functionality of R can be used to apply self-defined utility measures and store them in the '`sdcMicroObj`' object. Reports that summarize the anonymizations and their effect on the quality and risk of the data can be generated automatically.

Most methods are implemented efficiently in either C++ or by using package **data.table** for gaining computational speed especially for large data sets. The main functionalities of the package are illustrated here by applying them to a complex survey that is available with the package.

Computational details

All computations in this paper were performed using `Sweave` (Leisch 2002) with the following R session:

- R version 3.0.2 (2013-09-25), x86_64_w64-mingw32;
- base packages (R Core Team 2013): **base**, **datasets**, **graphics**, **grCevices**, **methods**, **stats**, **tools**, **utils**;
- **sdcMicro**, version 4.6.0 (Templ *et al.* 2015);

- **brew** (Horner 2011), **knitr** (Yihui 2013), **data.table** (Dowle and Short 2013), **xtable** (Dahl 2013), **MASS** (Venables and Ripley 2002), **Rcpp** (Eddelbuettel and Francois 2011), **sets** (Meyer and Hornik 2009).

Acknowledgments

sdcMicro has been supported by the International Household Survey Network (IHSN), see http://www.ihsn.org/home/software/disclosure-control-toolbox and also `vignette("sdc_guidelines")` in R; PARIS21 Secretariat at the Organisation for Economic Co-operation and Development (OECD; http://paris21.org/) and the Worldbank Development Data Group (http://data.worldbank.org/about/development-data-group).

References

(????). doi:10.1080/00401706.1999.10485670.

Alfons A, Kraft S (2013). ***simPopulation**: Simulation of Synthetic Populations for Surveys Based on Sample Data*. R package version 0.4.1, URL http://CRAN.R-project.org/package=simPopulation.

Alfons A, Kraft S, Templ M, Filzmoser P (2011). "Simulation of Close-to-Reality Population Data for Household Surveys with Application to EU-SILC." *Statistical Methods & Applications*, **20**(3), 383–407. doi:10.1007/s10260-011-0163-2.

Alfons A, Templ M (2013). "Estimation of Social Exclusion Indicators from Complex Surveys: The R Package **laeken**." *Journal of Statistical Software*, **54**(15), 1–25. doi:10.18637/jss.v054.i15.

Brand R (2002). "Microdata Protection Through Noise Addition." In *Inference Control in Statistical Databases*, Lecture Notes in Computer Science, pp. 97–116. Springer-Verlag.

Burridge J (2003). "Information Preserving Statistical Obfuscation." *Statistics and Computing*, **13**(4), 321–327. doi:10.1023/a:1025658621216.

Dahl D (2013). ***xtable**: Export Tables to LaTeX or HTML*. R package version 1.7-1, URL http://CRAN.R-project.org/package=xtable.

Domingo-Ferrer J, Torra V (2001). "A Quantitative Comparison of Disclosure Control Methods for Microdata." In *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, pp. 111–134.

Dowle M, Short T (2013). ***data.table**: Extension of* `data.frame` *for Fast Indexing, Fast Ordered Joins, Fast Assignment, Fast Grouping and List Columns*. R package version 1.8.10; with contributions from S Srinivasan, A Lianoglou and R Saporta, URL http://CRAN.R-project.org/package=data.table.

Drechsler J (2011). *Synthetic Datasets for Statistical Disclosure Control: Theory and Implementation*, volume 201 of *Lecture Notes in Statistics*. Springer-Verlag, New York. doi:10.1007/978-1-4614-0326-5.

Eddelbuettel D, Francois R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Elliot M (2000). "DIS: A New Approach to the Measurement of Statistical Disclosure Risk." *Risk Management*, **2**(4), 39–48. doi:10.1057/palgrave.rm.8240067.

Fischetti M, Salazar-González J (2000). "Complementary Cell Suppression for Statistical Disclosure Control in Tabular Data with Linear Constraints." *Journal of the American Statistical Association*, **95**(451), 916–928. doi:10.1080/01621459.2000.10474282.

Franconi L, Polettini S (2004). "Individual Risk Estimation in μ-**Argus**: A Review." In J Domingo-Ferrer (ed.), *Privacy in Statistical Databases*, Lecture Notes in Computer Science, pp. 262–272. Springer-Verlag.

Gouweleeuw J, Kooiman P, Willenborg L, De Wolf PP (1998). "Post Randomisation for Statistical Disclosure Control: Theory and Implementation." *Journal of Official Statistics*, **14**(4), 463–478.

Gower J (1971). "A General Coefficient of Similarity and Some of Its Properties." *Biometrics*, **27**(4), 857–871. doi:10.2307/2528823.

Horner J (2011). ***brew**: Templating Framework for Report Generation*. R package version 1.0-6, URL http://CRAN.R-project.org/package=brew.

Hundepool A, de Wetering AV, Ramaswamy R, Franconi L, Polettini S, Capobianchi A, de Wolf PP, Domingo J, Torra V, Brand R, Giessing S (2008). *μ-Argus. User Manual*. Version 4.2.

Ichim D, Franconi L (2010). "Strategies to Achieve SDC Harmonisation at European Level: Multiple Countries, Multiple Files, Multiple Surveys." In *Privacy in Statistical Databases'10*, pp. 284–296.

Kowarik A, Templ M, Meindl B, Fonteneau F (2013). ***sdcMicroGUI**: Graphical User Interface for Package **sdcMicro***. R package version 1.0.3, URL http://CRAN.R-project.org/package=sdcMicroGUI.

Leisch F (2002). "`Sweave`: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), *Compstat 2002 – Proceedings in Computational Statistics*, pp. 575–580. Physica Verlag, Heidelberg.

Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007). "l-Diversity: Privacy Beyond k-Anonymity." *ACM Transactions on Knowledge Discovery from Data*, **1**(1). doi:10.1145/1217299.1217302.

Manning A, Haglin D, Keane J (2008). "A Recursive Search Algorithm for Statistical Disclosure Assessment." *Data Mining and Knowledge Discovery*, **16**(2), 165–196. doi:10.1007/s10618-007-0078-6.

Maronna R, Martin D, Yohai V (2006). *Robust Statistics: Theory and Methods*. John Wiley & Sons, New York. doi:10.1002/0470010940.

Mateo-Sanz J, Sebe F, Domingo-Ferrer J (2004). "Outlier Protection in Continuous Microdata Masking." In *Privacy in Statistical Databases*, volume 3050 of *Lecture Notes in Computer Science*, pp. 201–215. Springer-Verlag.

Matthews G, Harel O (2011). "Data Confidentiality: A Review of Methods for Statistical Disclosure Limitation and Methods for Assessing Privacy." *Statistics Surveys*, **5**, 1–71. doi:10.1214/11-ss074.

Meyer D, Hornik K (2009). "Generalized and Customizable Sets in R." *Journal of Statistical Software*, **31**(2), 1–27. doi:10.18637/jss.v031.i02.

Muralidhar K, Parsa R, Sarathy R (1999). "A General Additive Data Perturbation Method for Database Security." *Management Science*, **45**(10), 1399–1415. doi:10.1287/mnsc.45.10.1399.

Muralidhar K, Sarathy R (2006). "Data Shuffling – A New Masking Approach for Numerical Data." *Management Science*, **52**(2), 658–670. doi:10.1287/mnsc.1050.0503.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rinott Y, Shlomo N (2006). "A Generalized Negative Binomial Smoothing Model for Sample Disclosure Risk Estimation." In *Privacy in Statistical Databases*, Lecture Notes in Computer Science, pp. 82–93. Springer-Verlag.

Rubin D (1993). "Discussion: Statistical Disclosure Limitation." *Journal of Official Statistics*, **9**(2), 461–468.

Samarati P, Sweeney L (1998). "Protecting Privacy When Disclosing Information: k-Anonymity and Its Enforcement Through Generalization and Suppression." *Technical Report SRI-CSL-98-04*, SRI International.

Shlomo N (2010). "Releasing Microdata: Disclosure Risk Estimation, Data Masking and Assessing Utility." *Journal of Privacy and Confidentiality*, **2**(1), 73–91.

Skinner C (2009). "Statistical Disclosure Control for Survey Data." In D Pfeffermann, C Rao (eds.), *Handbook of Statistics – Sample Surveys: Design, Methods and Applications*, volume 29 (Part A), pp. 381–396. North Holland.

Skinner C, Holmes D (1998). "Estimating the Re-Identification Risk per Record in Microdata." *Journal of Official Statistics*, **14**(4), 361–372.

Sweeney L (2002). "k-Anonymity: A Model for Protecting Privacy." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **10**(5), 557–570. doi:10.1142/s0218488502001648.

Task Force on the Conference of European Statisticians (2007). "Managing Statistical Confidentiality & Microdata Access. Principles and Guidelines of Good Practice." *Technical report*, United Nations Economic Comissionn for Europe, New York and Geneva.

Templ M (2008). "Statistical Disclosure Control for Microdata Using the R Package **sdcMicro**." *Transactions on Data Privacy*, **1**(2), 67–85.

Templ M (2011a). "Comparison of Perturbation Methods on Pre-Defined Quality Indicators." In *Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality, Tarragona, Spain*. Tarragona.

Templ M (2011b). "Estimators and Model Predictions from the Structural Earnings Survey for Benchmarking Statistical Disclosure Methods." *Research Report CS-2011-4*, Department of Statistics and Probability Theory, Vienna University of Technology. URL http://www.statistik.tuwien.ac.at/forschung/CS/CS-2011-4complete.pdf.

Templ M, Filzmoser P (2014). "Simulation and Quality of a Synthetic Close-to-Reality Employer-Employee Population." *Journal of Applied Statistics*, **41**(5), 1053–1072. doi:10.1080/02664763.2013.859237.

Templ M, Kowarik A, Meindl B (2014). "**sdcMicro** Case Studies." *Research Report CS-2014-1*, Department of Statistics and Probability Theory. Vienna University of Technology.

Templ M, Kowarik A, Meindl B (2015). ***sdcMicro**: Statistical Disclosure Control Methods for the Generation of Public- And Scientific-Use Files. Manual and Package*. R package version 4.6.0, URL http://CRAN.R-project.org/package=sdcMicro.

Templ M, Meindl B (2008a). "Robust Statistics Meets SDC: New Disclosure Risk Measures for Continuous Microdata Masking." In *Privacy in Statistical Databases*, volume 5262 of *Lecture Notes in Computer Science*, pp. 113–126. Springer-Verlag.

Templ M, Meindl B (2008b). "Robustification of Microdata Masking Methods and the Comparison with Existing Methods." In *Privacy in Statistical Databases*, volume 5262 of *Lecture Notes in Computer Science*, pp. 177–189. Springer-Verlag.

Templ M, Meindl B (2010). "Practical Applications in Statistical Disclosure Control Using R." In J Nin, J Herranz (eds.), *Privacy and Anonymity in Information Management Systems*, Advanced Information and Knowledge Processing, pp. 31–62. Springer-Verlag.

Venables W, Ripley B (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York. doi:10.1007/978-0-387-21706-2.

Yihui X (2013). ***knitr**: A General Purpose Package for Dynamic Report Generation in R*. R package version 1.5, URL http://CRAN.R-project.org/package=knitr.

Affiliation:

Matthias Templ
Department of Statistics and Probability Theory
Vienna University of Technology
Wiedner Hauptstr. 8–10
A-1040 Vienna, Austria
E-mail: templ@tuwien.ac.at

Alexander Kowarik, Bernhard Meindl, Matthias Templ
Methods Unit
Statistics Austria
Guglgasse 6
A-1110 Vienna, Austria
URL: http://www.statistik.at/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2015, Volume 67, Issue 4 *Submitted:* 2014-01-23
doi:10.18637/jss.v067.i04 *Accepted:* 2014-12-11