Matthias Templ

Statistical Disclosure Control for Microdata

Methods and Applications in R

Springer

Statistical Disclosure Control for Microdata

Matthias Templ

Statistical Disclosure Control for Microdata

Methods and Applications in R

Springer

Matthias Templ
Institute of Data Analysis and Process Design (IDP)
School of Engineering (SoE)
Zurich University of Applied Sciences (ZHAW)
Winterthur, Switzerland

and

data-analysis OG
Vienna, Austria
www.data-analysis.at

ISBN 978-3-319-50270-0 ISBN 978-3-319-50272-4 (eBook)
DOI 10.1007/978-3-319-50272-4

Library of Congress Control Number: 2017937274

Mathematics Subject Classification (2010): 62D99, 62D05, 62-07, 62J05, 62P25

© Springer International Publishing AG 2017
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To all who raised my interest in writing this book by not believing in our work and the success of free and open-source philosophy.

Preface

Introduction

An increasing amount of data on persons and establishments are collected by statistical organizations. Also, the demand for microdata for researchers is increasing since economic or empirical analysis, and to make statements about our society on empirical basis is often only possible when investigating in data with detailed information. Moreover, a considerable increase in the production of socioeconomic data and their accessibility by researchers have been observed in recent years. Statistical agencies are making more of their survey and census microdata available, government agencies are publishing more of their administrative data, and the private sector has become a major provider of big data.

This, however, comes with a variety of legal, ethical, and technical challenges. Privacy protection principles and regulations impose restrictions on access and use of individual data. Statistical producers are faced with the challenge of ensuring respondents' confidentiality when making microdata files accessible. Not only data producers are obligated to protect confidentiality, but also confidentiality is crucial for maintaining the trust of respondents and ensuring the honesty and validity of their responses. Statistical disclosure control (SDC) is thus an emerging field of research. Proper and secure microdata dissemination requires statistical agencies to establish policies and procedures that formally define the conditions for accessing microdata (Dupriez and Boyko 2010) and to apply statistical disclosure control (SDC) methods to data before release.

Also, the demand for complex microdata for training purposes for students seems to increase. Research projects under the 5th, 6th, or 7th framework program for research of the European Union generated confidential microdata for public or scientific use. Public-use files are also generated for researchers to be able to run complex simulation studies to compare methods. The *BLUE-ETS*, *AMELI*, *DACSEIS*, and *EUREDIT* research projects of this framework program are examples using anonymized data for simulation tasks. Outside Europe, there is a long tradition to anonymize data sets. For example, the US CENSUS Bureau provides

public-use data since the 1970s, and current releases of public-use data set include 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 addition, almost every statistical agency in the world and institutions holding confidential data, e.g., on health statistics, are at least interested in providing public- or scientific-use files of high quality and low disclosure risk. In any case, due to national laws on privacy, microdata cannot be distributed to the public or to researchers whenever re-identification of persons or establishments is possible.

This book gives a detailed view on well-known SDC methods for data, complex sample surveys, censuses, and administrative sources. It discusses the traditional approach of data anonymization by perturbation of the data, the disclosure risk, and data utility of anonymized data sets. It also includes methods to simulate synthetic data.

This is not a book about the software environment `R` and related packages for statistical disclosure control. The aim of the book was to explain the theory of statistical disclosure control methods. However, in order to better understand the theory, a lot of practical examples are given. Almost every example can be reproduced by the readers using `R` and the `R` packages **sdcMicro** and **simPop**. The code for each example is included in the book and provided as supplementary files at

http://www.statistik.tuwien.ac.at/public/templ/sdcbook

The free and open-source software provided with this book allows the reader to also apply the presented methods on their own data sets provided by the mentioned software.

Overview of the Book

This book is intended for statistical producers at National Statistical Offices (NSOs), data holders, and international and national agencies, as well as data users who are interested in the subject. It does not require no prior knowledge of statistical disclosure control (SDC). This book is focused on SDC methods for microdata. It does not cover SDC methods for protecting tabular outputs [see Castro 2010, for more details].

Chapter 1 includes a very brief introduction to R. It is intended to make further `R` code in the book understandable, but it does not replace a conventional introduction to R. If readers want to learn `R` in a broader context, we refer to the official manuals on `R` on the CRAN Webpage or to further contributed documentation on CRAN or books about R. After this short introduction, SDC tools and the differences between each of them are discussed. This section closes with a general discussion about the `R` package **sdcMicro**. The **sdcMicro** and the **simPop** packages are applied in various other chapters, especially in the case studies of Chap. 8.

The methods' parts start with an introduction to the basic concepts regarding statistical disclosure in Chap. 2. These concepts include the definition of terms used in the area of SDC and briefly present different kinds of disclosure scenarios. Chapter 2 finishes with risk-utility maps that show the trade-off between disclosure risk and data utility.

Chapter 3 includes methods for measuring disclosure risk. The aim is not only to discuss one concept of measuring/estimating disclosure risk, but also to describe several concepts. The chapter starts with basic methods such as *k*-anonymity and *l*-diversity. To measure risk on subsets of key variables, the SUDA method is explained. A large part of this chapter is dedicated to measuring the individual risk and the global risk with log-linear models.

In Chap. 4, the most common SDC methods are presented. The chapter is basically divided into two parts: one for categorical data and the other for continuous scaled data. Not only popular, but also new methods, such as shuffling, are discussed.

An introduction to common approaches for assessing information loss and data utility is given in Chap. 5. This chapter includes more than just typical comparisons, such as comparing means and covariances, since these methods—even regularly applied—are often too general and not suitable for specific data sets. Therefore, further concepts based on estimating certain indicators are presented as well.

Even if this book is mainly focused on traditional methods for SDC, Chap. 6 is dedicated to the simulation of synthetic data sets. A lot of different concepts exist in literature, but we focus on only one particular approach. We point out some advantages of the approach of model-based simulation of synthetic population data and describe the framework of this concept.

Chapter 7 provides practical guidelines on how to implement SDC on data sets in practice. Basically, the work flow for anonymizing data is shown. All the learned lessons are included in this workflow.

This workflow is also applied in Chap. 8 on several very popular data sets such as the Structure of Earnings Statistics (SES), the European Statistics on Income and Living Conditions (EU-SILC), the Family Income and Expenditure Survey (FIES), the International Income Distribution Database (I2D2), the Purchase Power Parity Data set (P4), and the Survey-based Harmonized Indicators Data (SHIP) that are also harmonized data files from household surveys.

Note that almost all methods introduced in this book can be implemented using **sdcMicroGUI**, an `R` -based, user-friendly, point-and-click graphical user interface (Kowarik et al. 2013). However, all methods are shown in applications using the `R`-package **sdcMicro** (Templ et al. 2015) that differs in that sense that working with the package is command line only and more methods are included than in **sdcMicroGUI** . Note that a successor, a new version of **sdcMicroGUI** that works in a browser and is included in **sdcMicro**, version > 4.7. The app can be opened via the function call `sdcApp()` in R. For simulating synthetic data, the package

simPop is used. Readers are encouraged to explore the implementation of methods using this book along with the detailed user manuals of **sdcMicroGUI** (Kowarik et al. 2013), **sdcMicro** (Templ et al. 2015), and **simPop** (Templ et al. 2017).

Winterthur, Switzerland Matthias Templ
March 2017

References

Castro, J. (2010). Statistical disclosure control in tabular data. In J. Nin & J. Herranz (Eds.), *Privacy and anonymity in information management systems, Advanced information and knowledge processing* (pp. 113–131). London: Springer.

Dupriez, O., & Boyko, E. (2010). Dissemination of microdata files. *Formulating policies and procedures*. IHSN Working Paper No 005, International Household Survey Network, Paris.

Kowarik, A., Templ, M., Meindl, B., & Fonteneau, F. (2013). *sdcMicroGUI: Graphical user interface for package sdcMicro*. URL:http://CRAN.R-project.org/package=sdcMicroGUI. *R* package version 1.1.1.

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 Commission for Europe, New York and Geneva.

Templ, M., Meindl, B., & Kowarik, A. (2015). Statistical disclosure control for micro-data using the *R* package sdcMicro. *Journal of Statistical Software*, *67*(1), 1–37.

Templ, M., Meindl, B., Kowarik, A., & Dupriez, O. (2017). Simulation of synthetic complex data: The R-package simPop. *Journal of Statistical Software,* 1–38. Accepted for publication in December 2015.

Acknowledgements

The author benefited from collaborations and discussions with **Bernhard Meindl** (Statistics Austria, data-analysis OG) and **Alexander Kowarik** (Statistics Austria, data-analysis OG). Many publications on this topic were jointly made and important input on parts of the book was given by them, especially on Chap. 8 but this is also true for some other chapters. The book also profited from previous work with Shuang Chen (OECD/Paris 21), Olivier Dupriez (World Bank), François Fonteneau (OECD/PARIS21), Thijs Benschop and Marius Totter (Vienna University of Technology). I would also like to thank the reviewers, especially Reviewer 1, for the numerous comments and suggestions. For English proofreading of parts of this book, my gratitude goes to Karin Zorn. The underlying software to this book was developed with financial support from the International Household Network, the World Bank (several projects, mentioned explicitly in the references), Google, Statistics Austria, data-analysis OG, and Vienna University of Technology.

The programming of the software was carried out in cooperation with Alexander Kowarik and Bernhard Meindl. Without their great contribution and professional work, the software would not be as powerful as it is at this stage. The code on model-based global risk measures was revised by Marius Totter while he was working on his master thesis, which I supervised. I would also like to thank my contact person and editor at Springer, Veronika Rosteck. She was always supportive and constructive, but most importantly, she was/is always in a good mood which motivated me to keep writing this book. Finally, my gratitude goes also to my wife Barbara for her supportive and understanding smile related to more intensive writing phases of the book.

Contents

Acronyms

CRAN	Comprehensive R Archive Network
CSS	Complex Survey Samples
EU-SILC	European Statistics on Income and Living Conditions
FIES	Family Income and Expenditure Survey
GUI	Point-and-click graphical user interface
I2D2	International Income Distribution Database
IHSN	International Household Survey Network
MCD	Minimum Covariance Determinant
MDAV	Maximum Distance to Average Vector
NSI	National Statistical Institute
P4	Purchase Power Parity Data
PRAM	Post Randomization Method
RMD	Robust Mahalanobis Distance
ROMM	Random Orthogonal Matrix Masking
SDC	Statistical Disclosure Control
SES	Structural Earning Statistics
SHIP	Survey-Based Harmonited Indicator Program
SUDA	Special Uniqueness Detection Algorithm

The book does not contain a glossary on Statistical Disclosure Control, because the important keywords are explained in Chap. 2. Moreover, glossaries on this topic can be found on the Internet, e.g., at http://neon.vb.cbs.nl/casc/Glossary.htm.

Chapter 1
Software

Abstract The methods used in this book are exclusively available in the software environment R (R Development Core Team 2014). A very brief introduction to some functionalities of R is given. This introduction does not replace a general introduction to R but shows some points that are important in order to understand the examples and the R code in the book. The package **sdcMicro** (Templ et al. 2015) forms a basis for this book and it includes all presented SDC methods. It is free, open-source and available from the comprehensive R archive network (CRAN). 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, microaggregation, adding correlated noise, shuffling and various other methods are integrated. With the package **sdcMicro**, statistical disclosure control methods can be applied in an exploratory, interactive and user-friendly manner. All results are saved in a structured manner and these results are updated automatically as soon a method is applied. Print and summary methods allow to summarize the status of disclosure risk and data-utility as well as reports can be generated in automated manner. In addition, most applications/anonymizations can be carried out with the point-and-click graphical user interface (GUI) **sdcMicroGUI** (Kowarik et al. 2013) without knowledge in the software environment R or the newer version of **sdcMicroGUI**, an app that is available within the package **sdcMicro** as function `sdcApp`. The new version runs in a browser and is based on **shiny** (Chang et al. 2016). A software package with a similar concept as **sdcMicro**—the **simPop** package (Templ et al. 2017)—is used to generate synthetic data sets.

1.1 Prerequisites

R can be used as an overgrown calculator. All operations of an calculator can be very easily used also in R, e.g. addition is done with `+`, subtraction with `-`, division with `/`, exponential with `exp()`, logarithm with `log()`, square root `sqrt()`, sinus `sin()`, …... All operations work as expected, e.g. the following expression is parsed by R, inner brackets are solved first, multiplication and division operators have precedence over the addition and subtraction operators, etc.

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_1

```
5 + 2 * log(3 * 3)

   ## [1] 9.394449
```

R is a function and object-oriented language. Functions can be applied to objects. The syntax is as shown in the following example

```
mean(rnorm(10))

   ## [1] -0.2763877
```

With the function `rnorm` 10 numbers are drawn from a standard normal distribution. Afterwards the `mean` is calculated for these 10 numbers. Functions typically have function arguments that can be set. The syntax for calling a function is in general

```
res1 <- name_of_function(v1) # an input argument
res2 <- name_of_function(v1, v2) # two input arguments
res3 <- name_of_function(v1, v2, v3) # three input arguments
# ...
```

Functions often have additional function arguments with default values You get access to all function arguments with `args()`

```
require(sdcMicro)
args(addNoise)

   ## function (obj, variables = NULL, noise = 150, method = "additive",
   ##     ...)
   ## NULL
```

The help for a function can be found with

```
?addNoise
```

Allocation to objects are made by <- or = and the generated object can be printed via object name followed by typing `ENTER`

```
x <- rnorm(10)
x

   ##  [1]  1.1908289  0.4773820 -1.0320670  2.5720002 -0.3302985
   ##  [6]  0.1175549  0.7655485 -0.4642652  0.3261664  0.1262540
```

Please note that R is case sensitive.

1.1.1 Installation and Updates

If R is already installed on the computer, ensure that you work with the current version of R. If the software is not installed, go to http://cran.r-project.org/bin/ and choose your platform. For Windows, just download the executable file and follow the on-screen instructions.

1.1.2 Install sdcMicro and Its Browser-Based Point-and-Click App

Open R on your computer and type:

```
install.packages("sdcMicro")
install.packages("simPop")
```

Installation is needed only once. Note that (only) the GUI requires the GTK+ package to draw windows. When installing **sdcMicroGUI** (`install.packages("sdcMicroGUI")`), all required packages (including GTK+) are automatically installed if the user have sufficient system administration rights. From version 4.7.0 onwards, the GUI is included in package **sdcMicro** and started with `sdcGUI()`. From version 5.0.0 onwards the GUI is replaced by a shiny app. The browser-based app can be started via

```
sdcApp()
```

1.1.3 Updating the SDC Tools

Typing `update.packages()` into R searches for possible updates and installs new versions of packages if any are available. For the **sdcMicroGUI**, users can also click on the menu item *GUI* → *Check for Updates*; this should be done regularly.

The previous information was related to install the stable CRAN version of the packages. However, latest changes are only available in the development version of the package. This 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.

To install the latest development version, the installation of the package **devtools** (Wickham and Chang 2015) is needed. After calling the **devtools** package, the development versions can be installed via `install_github()`

```
require("devtools")
install_github("alexkowa/sdcMicro")
```

1.1.4 Help

It is crucial to have basic knowledge about getting help. With

```
help.start()
```

your browser opens and help (and more) is available.

The clickable help index of the package can be accessed by typing the following command into R.

```
help(package=sdcMicro)
```

To find a specific help of a function, one can use `help(name)` or `?name`. As an example, we look at the help file of function `rankSwap`, which is included in the package **sdcMicro**

```
library(sdcMicro)
?rankSwap
```

Data in the package can be loaded via the `data()` function.

```
data(testdata)
```

`help.search()` can be used to find functions for which you don't know its exact name, e.g.

```
help.search("adding noise")
```

will search your local R installation for functions matching the character string `"adding noise"` in the (file) name, alias, title, concept or keyword entries. With function `apropos` one can find and list objects by (partial) name. For example, to list all objects with partial name match `risk`:

```
apropos("risk")

   ## [1] "calcRisks"        "dRisk"            "dRiskRMD"
   ## [4] "indivRisk"        "LLmodGlobalRisk"  "measure_risk"
   ## [7] "modRisk"          "risk"
```

It can be seen that some useful function names such as `measure_risk` or `calcRisks` are listed.

To search help pages, vignettes or task views, using the search engine at http://search.r-project.org and to view them in your web browser, you can use

```
RSiteSearch("adding noise")
```

which reports all search results for the character string `"adding noise"`.

1.1.5 The R Workspace and the Working Directory

Created objects are available in the workspace of R and loaded in the memory of your computer. The collection of all created objects is called *workspace*. To list the objects in the workspace

```
ls()

   ## character(0)
```

When importing or exporting data, the working directory must be defined. To show the current working directory, the function `getwd` can be used

```
getwd()

   ## [1] "/Users/teml/workspace/sdc-springer/book"
```

To change the working directory, the function `setwd` is the choice

```
# paste creates a string
p <- paste(getwd(), "/data", sep="")
p

   ## [1] "/Users/teml/workspace/sdc-springer/book/data"

# now change the working directory
setwd(p)
# has it changed? Yes...
getwd()

   ## [1] "/Users/teml/workspace/sdc-springer/book/data"
```

1.1.6 Data Types

The objective is to know the most important data types: numeric, character and logical as well as the data structures

- vectors/factors
- lists
- data frames
- special data types: missing values, NULL-objects, NaN, $-/+$ Inf

Vectors are the simplest data structure in R. A vector is a sequence of elements of the same type such as numerical vectors, character vectors, logical vectors Vectors are often created with the function `c()`, e.g.

```
v.num <- c(1,3,5.9,7)
v.num

   ## [1] 1.0 3.0 5.9 7.0

is.numeric (v.num)

   ## [1] TRUE
```

`is.numeric` query if the vector is of class numeric. Note that characters are written with parenthesis.

Logical vectors are often created indirectly from numerical/character vectors

```
v.num > 3

   ## [1] FALSE FALSE  TRUE  TRUE
```

Many operations on vectors are performed element-wise, e.g. logical comparisons or arithmetic operations with vectors. A common error source is when the length of vectors differ. Then the shorter one is repeated (*recycling*)

```
v1 <- c(1,2,3)
v2 <- c(4,5)
v1 + v2

   ## [1] 5 7 7
```

One should also be aware that R coerces internally to meaningful data types automatically. For example,

```
v2 <- c (100, TRUE, "A", FALSE)
v2

   ## [1] "100"   "TRUE"  "A"     "FALSE"

is.numeric (v2)

   ## [1] FALSE
```

Here the lowest common data type is a string and therefore all entries of the vector are coerced to character. Note, to create vectors, the functions `seq` and `rep` are very useful.

Often it is necessary to subset vectors. The selection is made using the `[]` operator. A selection can be done in three ways

positive: a vector of positive integers that specifies the position of the desired elements

negative: a vector with negative integers indicating the position of the non-required elements

logical: a logic vector in which the elements are to be the selected (TRUE), and those who are not selected (FALSE).

```
require("laeken")
data("eusilc")
# extract for 10 observations of variable age from eusilc data
age <- eusilc[1:10, "age"]
age

   ##  [1] 34 39  2 38 43 11  9 26 47 28

# positive indexing:
age[c(3,6,7)]

   ## [1]  2 11  9

# negative indexing:
age[-c(1,2,4,5,8:10)]

   ## [1]  2 11  9

# logical indexing:
age < 15

   ##  [1] FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE

# a logical expression can be written directly in []
age[age < 15]

   ## [1]  2 11  9
```

A list in R is a *ordered* collection of objects whereas each object is part of the list and where the data types of the individual list elements can be different (vectors, matrices, data.frames, lists, etc.). The dimension of each list item can be different. Lists can be used to group and summarize various objects in an object. There are (at least) three ways accessing elements of a list, (a) the `[]` -operator, the operator `[[]]`, the `$` -operator and the name of a list item. With `str()`), you can view the structure of a list, with `names()` you get the names of the list elements

```
## measure risk on data frames
data(Tarragona)
x <- Tarragona[, 5:7]
## add noise
y <- addNoise(x)
## estimate the risk, result is a list
risk <- dRiskRMD(x, xm=y$xm)
class(risk)

   ## [1] "list"

str(risk)

   ## List of 8
   ##  $ risk1     : num 0.0048
   ##  $ risk2     : num 0.0048
   ##  $ wrisk1    : num 0.0117
   ##  $ wrisk2    : num 0.0117
   ##  $ indexRisk1: Named int [1:4] 154 160 744 812
   ##   ..- attr(*, "names")= chr [1:4] "154" "160" "744" "812"
   ##  $ indexRisk2: int [1:4] 154 160 744 812
   ##  $ riskvec1  : num [1:834] 0 0 0 0 0 0 0 0 0 0 ...
   ##  $ riskvec2  : num [1:834] 0 0 0 0 0 0 0 0 0 0 ...

names(risk)

   ## [1] "risk1"      "risk2"      "wrisk1"     "wrisk2"
   ## [5] "indexRisk1" "indexRisk2" "riskvec1"   "riskvec2"

## access elements from the named list
risk$wrisk2

   ## [1] 0.01172644
```

Factors in R are of special importance. They are used to represent nominal or ordinal data. More precisely, unordered factors for nominally scaled data and ordered factors for ordinal scaled data. Factors can be seen as special vectors. They are internally coded integers from 1 to n (# of occurrences) which are all associated with a name (label). So when and why variables should be stored as class factor? Basically, factors has to be used for categorical information to get the correct number of degrees of freedom and correct design matrices in statistical modeling. In addition, the implementation of graphics for factors versus numerical/character vectors differ. Moreover, factors are more efficient in storing of character vectors However, factors have a more complex data structure since factors include a numerically coded data vector and labels for each level/category.

```
## access a vector with the dollar operator
gender <- eusilc$rb090
## first six values:
head(gender)

   ## [1] female male   male   female male   male
   ## Levels: male female
```

Data frames (in R `data.frame`) are the most important data type. This corresponds to the well-known from other software packages rectangle data format with *rows* correspond to observation units and *columns* to variables. A `data.frame` is like a `list` whereas all list elements are vector/factors but with the restriction that all list elements have the same number of elements (equal length). For example, data from external sources to be read are often stored as data frames, i.e. data frames are usually created by reading data but they can also be constructed with function `data.frame()`.

A lot of opportunities exist to subset a data frame, e.g. with syntax: `[`*index row*`,` *index columns*`]`. Again positive, negative and logical indexing is possible and the type of indexing may be different for row index and column index. To access to individual columns is most easy with the `$` -operator (like lists).

```
## babies of age 1 living in households of size 2:
babies1 <- eusilc$age == 1 & eusilc$hsize == 2
str(babies1)

   ##  logi [1:14827] FALSE FALSE FALSE FALSE FALSE FALSE ...

## select some variables, e.g. including
## family/children related allowances
cn <- colnames(eusilc) %in% c("rb090", "db040", "hy050n")
str(cn)

   ##  logi [1:28] FALSE FALSE TRUE FALSE FALSE TRUE ...

eusilc[babies1, cn]

   ##        db040  rb090  hy050n
   ## 6333  Styria female 2009.54
   ## 13860 Vienna   male 1897.37

## or for short:
eusilc[eusilc$age == 1 & eusilc$hsize == 2, c("rb090", "db040", "hy050n")]

   ##        rb090  db040  hy050n
   ## 6333  female Styria 2009.54
   ## 13860   male Vienna 1897.37
```

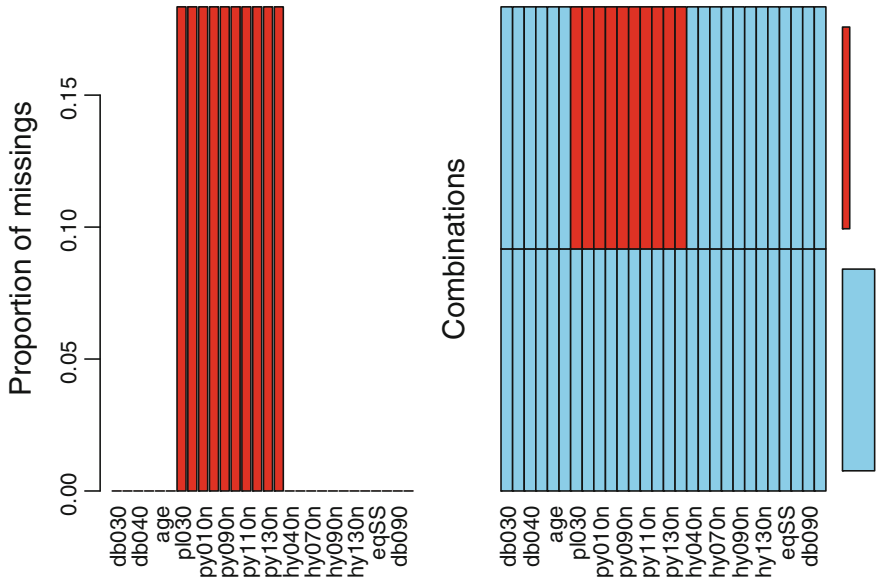


Fig. 1.1 Missing patterns in the EU-SILC data set. *Left* proportion of missing values in each variable. *Right* Missing value patterns

A few helpful functions that can be used in conjunction with data frames are `dim()`, reporting the dimension (number of rows and columns), `head()`, the first (default 6) rows of a data frame, `colnames()`, the columns/variable names

Missing values are almost always present in the data. The default representation of missing value in R is the symbol `NA`. A very useful function to check if data values are missing is `is.na`. It returns a logical vector or data.frame depending if the input is a vector or data.frame indicating missingness. To calculate the number of missing values, we could sum the TRUE's (interpreted as 1 while FALSE is interpreted as 0).

```
sum(is.na(eusilc))

   ## [1] 27200
```

All in all, 27200 values are missing, this is approx. 6.55% of the data values.

```
27200 / (nrow(eusilc) * ncol(eusilc)) * 100

   ## [1] 6.551754
```

To analyse the structure of missing values, the R package **VIM** (Templ et al. 2012). For the EU-SILC data set we immediately see the source of missingness, see Fig. 1.1.

```
require("VIM")
aggr(eusilc)
```

More precisely, we see the monotone missingness in the personal income components of the European Union Statistics on Income and Living Conditions (EU-SILC) data set. The proportion of missings in this variables is almost 20% (see left plot). In the right plot of Fig. 1.1 the patterns of missingness are visible. Most of the observations have no missings, the rest have missing values in each of the personal income components. In fact, these are children that have no personal income.

Before applying SDC methods it is recommended to look at the structure of missingness since missing values has effect on risk measurement. More information on missing values can be found in Templ et al. (2012).

1.1.7 Generic Functions, Methods and Classes

These topics are only discussed very briefly since they are more advanced. However, since **sdcMicro** is highly object-oriented some basics are good to know. The use of object-orientation makes implementations in R very user-friendly. First we replicate some basic concepts about classes in R.

R has different class systems, the most important ones are S3 and S4 classes. Programming with S3 classes is lazy living, it is simpler to program in S3 than in S4. However, S4 is more *clean* and the use of S4 can make packages very user-friendly.

In any case, in R each object is assigned to a class (attribute *class*) Classes allow object-oriented programming and *overloading* of *generic functions*. Generic functions produce different output for objects different classes as soon as methods are written for such classes.

This sounds complex, but with the following example it should get clearer.

As an example of a generic function, we use the function `summary`. `summary` is a generic function used to produce result summaries. The function invokes particular methods which depend on the class of the first argument.

```
## how often summary is overloaded with methods
## on summary for certain classes
length(methods(summary))

   ## [1] 175

class(eusilc$hsize)

   ## [1] "integer"

summary(eusilc$hsize)

   ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   ##   1.000   2.000   3.000   3.234   4.000   9.000

## convert it to a factor
eusilc$hsize <- as.factor(eusilc$hsize)
class(eusilc$hsize)

   ## [1] "factor"

summary(eusilc$hsize)

   ##    1    2    3    4    5    6    7    8    9
   ## 1745 3624 3147 3508 1815  630  252   88   18
```

From this previous example one can see that the summary is different, depending on the class of the object. R internally looks if a method is implemented for the given class of the object. If yes, this function is used, if not, the function `summary.default` is used. This procedure is called *method dispatch*. In the previous example, last line, R looked if a function `summary.factor` is available, which was true.

Note that generic functions can be defined, and we also can define print, summary and plot methods for objects of certain classes.

As mentioned before, S4 classes are different and more formal. The important fact to know when working with **sdcMicro** is how to access elements of S4 class objects.

In the next code, a S4 class object of class *sdcMicroObj* is generated.

```
sdc <- createSdcObj(eusilc,
  keyVars=c('age','rb090','db040','hsize','pb220a'),
  numVars=c('eqIncome'), w='rb050')
class(sdc)

   ## [1] "sdcMicroObj"
   ## attr(,"package")
   ## [1] "sdcMicro"
```

The defined functions for S3 classes, such as `names()` or `head()`, does not longer work. For example, the replacement for `names()` is `slotNames()`

```
slotNames(sdc)
```

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

In addition, the $ operator do not longer work, but its S4 counterpart–the slot– must be used. For example to access the slot `risk` we use `sdc@risk` This slot contains a list and this list contains again a list, that can be also accessed

```
str(sdc@risk)
```

```
## List of 3
##  $ global    :List of 5
##   ..$ risk     : num 0.00224
##   ..$ risk_ER  : num 33.1
##   ..$ risk_pct : num 0.224
##   ..$ threshold: logi NA
##   ..$ max_risk : num 0.01
##  $ individual: num [1:14827, 1:3] 0.001961 0.012359 0.000495 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "risk" "fk" "Fk"
##  $ numeric   : num 1
```

```
str(sdc@risk$global)
```

```
## List of 5
##  $ risk     : num 0.00224
##  $ risk_ER  : num 33.1
##  $ risk_pct : num 0.224
##  $ threshold: logi NA
##  $ max_risk : num 0.01
```

```
sdc@risk$global$risk_pct
```

```
## [1] 0.2235023
```

To make the life easier, there are accessor functions defined for working with objects of class `sdcMicro`, see Sect. 1.4.3.

1.2 Brief Overview on SDC Software Tools

μ-Argus

Under the 5-th framework programme of the European Union, a general tool for the anonymization of micro-data, μ-**Argus**, has been improved. The software is still developed by Statistics Netherlands and other partners, and some of those extensions are being subsidized by Eurostat. It features a graphical point and click user interface, which was based on Visual Basic until version 4.2. and is now (Version 5.1 and onwards) written using Java and can be downloaded for free from the CASC website (http://neon.vb.cbs.nl/casc/mu.htm). Currently, only 32-bit versions have been built, and there is no command line interface available.

C++ Code from the International Household Survey Network

Previous efforts from the International Household Survey Network (IHSN) include the development of microdata anonymization software. IHSN developed `C++` code for microdata anonymization in order to support the safe dissemination of confidential data. In addition, plug-ins were developed to call the code from statistical software, namely from `Stata`, `SPSS` and `SAS`. While the software developed from the IHNS is free and open-source, the use of the code from the previous mentioned statistical software is restricted for commercial use since the users have to buy a license for the statistical software. The IHSN code is fully integrated (and improved) in **sdcMicro**.

sdcMicro and sdcMicroGUI

Package **sdcMicroGUI** (Kowarik et al. 2013) provides a graphical user interface for **sdcMicro** and it serves as an easy-to-handle, highly interactive tool for users who want to use the **sdcMicro** package for statistical disclosure control but are not familiar with the native R command line interface. The GUI performs automated recalculation and display of frequency counts, individual and global risk measures, information loss and data utility after each anonymization step. Changes to risk and utility measurements of the original data are also conveniently displayed in the graphical user interface (GUI) and the code is saved in a script, which can easily be exported, modified and re-used, making it possible to reproduce any results. Currently, a new re-implementation of **sdcMicroGUI** will be made using **shiny** (Chang et al. 2016). The aim is that methods can be used in a browser. The package should be available in autumn 2016.

Other tools for data from biomedical sciences

Biomedical data sets have (usually) a simple structure, i.e. categorical data, few key variables, no complex designs as well as no hierarchical or cluster structures. Therefore, only simple tools for measuring the disclosure risk and simple tools for perturbing the data sets are in use.

With **TIAMAT** (Dai et al. 2009) different k-anonymization techniques can be compared. Also the **eCPC toolkit** from the Swedish eScience Research Center (SeRC) and the **UTD Anonymization ToolBox** developed by the Data Security

and Privacy Lab at the University of Texas at Dallas allows investigations in k-anonymity. They all are based on the Mondrian algorithm (LeFevre et al. 2006) to achieve k-anonymity. **AnonTool**, developed by the department of Department of Informatics Systems at the Alpen-Adria-Universität Klagenfurt, provides two methods: k-anonymity and l-diversity (see for both Sect. 3.3). All specifications are given by an XML file. **Arx** (Kohlmayer et al. 2012) is implemented in Java. The anonymization in **Arx** consists of three basic steps, first to configure the anonymization process. Second, to explore the so-called solution space and third, analyzing the perturbed data. The tool is useful for k-anonymity, l-diversity and similar approaches such as t-closeness or δ-presence (Nergiz et al. 2007). It provides interactive features to investigate in the information loss based on univariate summaries of the original and perturbed data.

1.3 Differences Between SDC Tools

Table 1.1 gives an overview of software and available methods of four concrete software products—the μ-**Argus** software (Hundepool et al. 2008) from Statistics Netherlands, the R packages **sdcMicro** (Templ et al. 2015) and **sdcMicroGUI** (Kowarik et al. 2013) and C++ code that was written by the IHSN (see http://www.ihsn.org/home/node/32). In addition, the tools available from the biomedical area (such as **Arx**) are summarized in the field "Biomed Tools".

From the "Biomed" tools, **Arx** is the most powerful one due to its well-designed point and click graphical user-interface. In addition, it has more methods integrated as other tools in the biomedical area. However, it cannot deal with data from complex designs and has limited features apart from frequency-based procedures.

The difference of μ-**Argus** and **sdcMicro** is not only with the amount of supported methods. The main advantages of **sdcMicro** is 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) and its flexibility to use. Note that reproducibility, a lot of interactive features (e.g., automatic code generation) and computational optimized code as well as automatic recalculations are not provided by μ-**Argus**. In addition, many methods are missing in μ-**Argus** and some methods are implemented in an oversimple manner. For example, SUDA, shuffling or model-based risk estimation are not available in μ-**Argus**. Or one very central method is local suppression with the aim to provide data that fulfills k-anonymity (see Sect. 3.3). When having for example seven key variables defined, in μ-**Argus** subsets of key variables (typically up to all combinations of four variables) are made k-anonym, but k-anonymity is usually not fulfilled for the given seven key variables. In **sdcMicro** this subset approach can be applied, but it is also possible to ensure k-anonymity for all seven variables. Since methods can not be applied using a command line interface in μ-**Argus** we can not compare computation speed of **sdcMicro** but we can state that μ-**Argus** is not suitable for large data sets. It becomes slow and

Table 1.1 List of methods supported by different statistical disclosure control software. Ticks in brackets indicate only limited implementation of a method. The table is originally published in Templ et al. (2015). Published with kind permission of ©Matthias Templ, Alexander Kowarik and Bernhard Meindl 2015 under the terms of the creative commons attribution license which permits any use, distribution, and reproduction in any medium provided the original author(s) and source are credited.

Method | Software	μ-**Argus** 4.2	**BioMed tools**	**sdcMicro** > 4.3.0	**sdcMicroGUI** > 1.1.0	**IIHSN**
frequency counts	✔	✔	✔	✔	✔
individual risk (IR)	✔		✔	✔	✔
IR on households	✔		✔	✔	✔
l-diversity		✔	✔	✔	✔
suda2			✔		✔
global risk (GR)	✔		✔	✔	✔
GR with log-lin mod.			✔		
recoding	✔	✔	✔	✔	(✔)
local suppression	(✔)	(✔), Arx: ✔	✔	✔	(✔)
swapping	(✔)		✔		✔
pram	✔		✔	✔	✔
adding correlated noise			✔	✔	✔
microaggregation	✔		✔	✔	✔
shuffling			✔	✔	
utility measures	(✔)	Arx: (✔)	✔	✔	
GUI	(✔)	✔		✔	
CLI		Arx: ✔	(✔)		✔
reporting	✔		✔	✔	
platform independent		Arx: ✔	✔	✔	✔
free and open-source		Arx: ✔	✔	✔	✔

runs out-of-memory even with medium-sized data sets while with **sdcMicro** huge data sets up to millions of observations can be processed.

An exercise in computation time is shown in Fig. 1.2. We investigate in the computation time of the most computer-intense methods, local suppression and risk measurement. **sdcMicro** and IHSN code is compared, other software either do not have a command line interface for comparison (so as μ-**Argus**), the tested procedures are not available (**Arx**) or the procedures are such slow (also μ-**Argus**) that it makes no sense to put it in this figure. Figure 1.2 the performance increase with respect to computation time of the current version of **sdcMicro**, 5.0.0 compared to the previous implementation in **sdcMicro** (<version 4.1.0) using IHSN C++ code is shown. To measure the computation time, the Bangladesh Survey on Income (I2D2) with 48969 observations on 41 variables was used. However, to increase the size of the data, the observations were randomly replicated to enlarge the data set up to

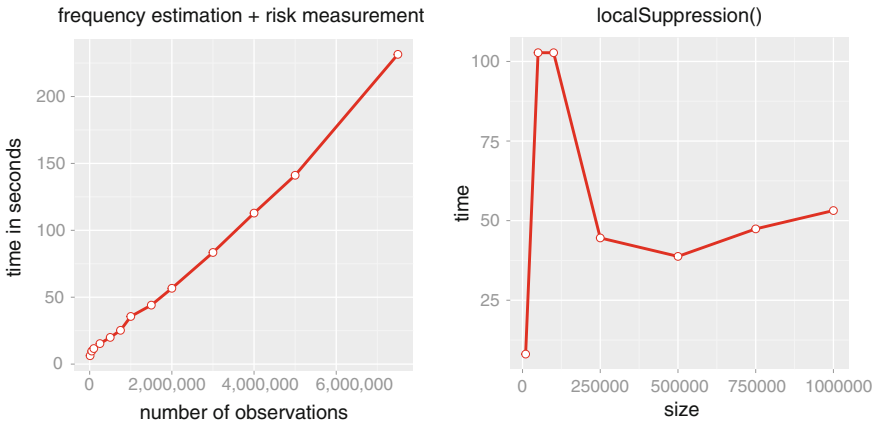


Fig. 1.2 Computation time of IHSN C++ code (equals sdcMicro version $< 4.1.0$) and sdcMicro (version $\geq 4.1.0$)

7,500,000 observations. For different numbers of observations, frequency counts and risk (left plot in Fig. 1.2) as well as (heuristic "optimal") local suppression (right plot in Fig. 1.2) were applied independently for each data set.

While for the IHSN C++ solutions the computation time is exponential regarding the number of observations, the new implementation features a "lower than linear growth" for local suppression. The decrease in computation time between 100.000 and 500.000 observations can be explained with the internal use of measuring k-anonymity (see Sect. 3.3) in the implementation of the local suppression method in **sdcMicro**. The size of the data set is reduced internally, because observations with frequencies larger k can be omitted from the analysis since they do not have any risk. Thus, the larger the data set the more observation can be omitted for calculations for this method.

The figures clearly show that the **sdcMicro** can also be applied on very large data sets. Of course, there are also some experimental methods included with much higher demand on computation time. However, the main features of **sdcMicro** are efficiently implemented.

1.4 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**. To open the GUI type

```
require("sdcMicroGUI")
sdcGUI()
```

This will load the **sdcMicroGUI** package into R and display the point-and-click GUI. If you have not installed **sdcMicroGUI**, you will see an error message; follow the steps described in Sect. 1.1.1 to install the package.

However, the GUI is not in focus of the book but it can be used to replace most of the command line instructions in this book just by point-and-click in the GUI.

1.4.1 General Information About sdcMicro

The first version, version 1.0.0, of the **sdcMicro** package was released in 2007 on the comprehensive R archive network (`http://cran.r-project.org`) 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 5.0.0, is a huge step forward. Almost all methods are implemented in an object-oriented manner (using S4 classes) and have been written internally either by implementations in C++ or by using the package **data.table** (Dowle et al. 2013). This allows for high-performance computations. The IHSN provided C++ code for many methods which were rewritten from scratch (except `suda2` and `rankSwap`) and integrated into **sdcMicro**.

1.4.2 S4 Class Structure of the sdcMicro Package

This section is mainly based on Templ et al. (2015) that shows the implementation of **sdcMicro** in detail. The following list gives an overview about the general aim of **sdcMicro** (see also Templ et al. 2015):

- **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 use its functionalities from **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 et al. 2013);
- dynamic reports about results of the anonymization process can be generated.

Complex survey data needs special attention, but also for data collected without complex sampling designs some information is crucial. Of course, the software tool must be told which variables are important to protect and–if available in the data–the

information about the variables holding the information on sampling weights and possible cluster structures (like household ID's) have to be reported. The idea is to generate an object in R that contains all these information, and in addition, stores further information that is estimated from the data. Moreover the information should update as soon a method is applied on such an object. To create such an object, the function `createSdcObj` is of central importance.

Of course, the add-on R package **sdcMicro** has to be loaded first in R to make the needed functions available. Parameters for `createSdcObj` 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 test data from a survey of the Philippines that are also included in the **sdcMicro** package.

```
require("sdcMicro")
data("testdata", package="sdcMicro")
sdc <- createSdcObj(testdata,
          keyVars=c('urbrur','water','sex','age'),
          numVars=c('expend','income','savings'),
          pramVars=c("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 Sect. 4.2.3), the vector of weights and the household IDs.

As mentioned before, by calling `createSdcObj` not only the previous mentioned information is stored, but many other slots are filled, e.g. slots holding information on risk and utility. Some of them are empty and only be filled as soon a certain method is applied, e.g. a slot called `pram` is only filled if the pram method is applied explicitly. The information contained in the slots update update when new methods applied on the object, e.g. the slot holding the information on risk and utility updates automatically with new risk and utility estimates as soon as an anonymization method is applied.

The following slots of an object of class `sdcMicroObj` are available:

```
slotNames(sdc)
```

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

Slot `origData` contains the original data, `keyVars` the index of categorical key variables. `pramVars` contains an index indicating variables that are pramed (see Sect. 4.2.3 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`. The risk measures (see Chap. 3) 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, local suppressions are stored in slots `pram` and `localSuppression` while additional results (e.g., self-defined utility measures) are saved in slot `additionalResults`. Optionally, under the slot `prev` previous results are saved. Wrapper functions to extract relevant information from the `sdcMicroObj` are available (see Sect. 1.4.3). For more details on the structure of the `sdcMicroObj` have a look at the help file (`help("createSdcObj")`).

The show (print) method shows some basic properties of objects of class `sdcMicroObj`. The output of this print method is discussed in Sect. 3.3.

```
print(sdc)

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 330 (7.205%)
   ##   - 3-anonymity: 674 (14.716%)
   ##   - 5-anonymity: 1288 (28.122%)
   ## ## ---------------------------------------------------------------------
```

Methods are applied on the `sdcMicroObj` 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` are updated. A method to an `sdcMicroObj` can be applied by

```
method(sdcMicroObj),
```

where `method` is a placeholder for a specific method.

We note that **sdcMicro** also supports the straightforward application of methods to micro-data. For example, microaggregation of three continuous key variables (*expend*, *income* and *savings*) on the data set `testdata` can be achieved with

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

  ##
  ##  Object created with method mdav and aggregation level 3
  ##  -------------------------
  ## x ... original values
  ##      expend             income             savings
  ##  Min.   :    3377   Min.   :2.897e+03   Min.   :   2975
  ##  1st Qu.:25610225   1st Qu.:2.510e+07   1st Qu.:2434823
  ##  Median :50462300   Median :5.075e+07   Median :4982921
  ##  Mean   :50499785   Mean   :5.012e+07   Mean   :4964039
  ##  3rd Qu.:75513585   3rd Qu.:7.500e+07   3rd Qu.:7487258
  ##  Max.   :99962044   Max.   :1.000e+08   Max.   :9997808
  ##
  ##  -------------------------
  ## mx ... microaggregated values
  ##      expend             income            savings
  ##  Min.   : 1151705   Min.   :  621527   Min.   :  85812
  ##  1st Qu.:25359582   1st Qu.:25066667   1st Qu.:2373826
  ##  Median :50285163   Median :50766667   Median :4990754
  ##  Mean   :50499785   Mean   :50115690   Mean   :4964039
  ##  3rd Qu.:75370230   3rd Qu.:75100000   3rd Qu.:7522210
  ##  Max.   :99174303   Max.   :99200000   Max.   :9928146
  ##
  ##  -------------------------
  ## Try names(your object from class micro) for more details
```

From the previous code line it is visible that an object of class *sdcMicroObj* is not mandatory to have. The methods can directly applied on data frames.

However, it is more convenient to first create an object of class *sdcMicroObj* using `createSdcObj()` and apply methods on objects of this class. To apply microaggregation not on a data frame but on such a class, we use

Table 1.2 Functions in R package **sdcMicro** for SDC disclosure risk and utility methods

Function	Aim	Updates slots ...
`freqCalc()`	Sample and population frequency estimation (used by `measureRisk()`)	–
`suda2()`	Frequency calculation on subsets	`@risk$suda2`
`ldiversity()`	*l*-diversity	`@risk$ldiversity`
`measureRisk()`	Individul, household and global risk estimation	`@risk*`
`modRisk()`	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$numericRMD` (risk on cont. variables)
`dUtility()`	Data utiltiy measures	`@utility$*`

```
sdc <- microaggregation(sdc)
```

with sdc being an object of class sdcMicroObj. Note that the continuous key variables are already defined via createSdcObj(). To speak in more abstract terms: a method be applied to an object of class *sdcMicroObj* will extract the needed information, apply the method and put back the results as well as it will update several slots (e.g. update risk estimates).

In Table 1.2, the currently available methods and its function calls are listed. A brief aim of the function as well as an information which slots get updated in the sdcMicroObj object are outlined (see also Templ et al. 2015).

Table 1.3 Functions in R package **sdcMicro** for anonyimzation/perturbation methods

Function	Aim	Updates slots ...
globalRecode()	Anonyization 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$*, @localSuppression, @manipKeyVars, @prev
kAnon()	Local suppression to achieve k-anonymity	@risk$*, @localSuppression, @manipKeyVars, @prev
pram()	Swapping values using the post randomisation method	@pram, @manipPramVars, @prev
microaggrGower()	Microaggregation 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

For example, the application of `localSuppression()` on an `sdcMicroObj` suppress 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 with new estimates and in the slot `localSuppression` the information of suppressed values gets updated. Another example is to apply `microaggregation`. In the above code, method microaggregation 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 microaggregation is applied on all continuous key variables. Finally, slot `@risk$numeric` (disclosure risk for continuous key variables) and `@utility$*` (data utility for continuous key variables) and `@manipNumVars` (the perturbed variables) are updated or filled. In addition, information on the previous state of the anonymization is saved in `@prev` (see Table 1.3).

1.4.3 Utility Functions

Available help functions of **sdcMicro** are listed in Table 1.4 (see also Templ et al. 2015). 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.

Table 1.4 Utility functions

Function	Aim
`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 return slots from `sdcMicroObj` objects. Alternatively, `slot` can also be used.
`generateStrata()`	Generate single variable defining a strata from multiple variables
`undolast()`	Revert 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()`	Change a key variable of an object of class `sdcMicroObj` from `numeric` to `factor` or from `factor` to `numeric`

The slots of the `sdcMicroObj` can be accessed also using function `get.sdcMicroObj()` or `slot` as well as the current state of the data (with all anonymizations done so far) can be extracted, see the following code where the data utility and the categorical key variables are extracted from the *sdcMicroObj* `sdc`.

```
## data utility:
ut <- slot(sdc, "utility")
## index of categorical key variables:
cat <- slot(sdc, "keyVars")
## key variables, orginal data
head(testdata[, cat], 3)

##   urbrur water sex age
## 1      2     3   1  46
## 2      2     3   2  41
## 3      2     3   1   9
```

Using the function `extractManipData`, the manipulated data can also be extracted from the object `sdc`.

```
dat <- extractManipData(sdc)
str(dat)

## 'data.frame': 4580 obs. of 15 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 3 3 3
...
## $ electcon : int 1 1 1 1 1 1 1 1 1 1 ...
## $ relat : int 1 2 3 3 1 2 3 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 : int 90929693 27338058 26524717 18073948 6713247 49057636
63386309 1106874 32659507 34347609 ...
## $ income : num 5.78e+07 2.53e+07 6.92e+07 7.96e+07 9.03e+07 ...
## $ savings : num 116258 279345 5495381 8695862 203620 ...
## $ 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 ...
## $ household_weights: num 25 25 25 25 16.7 ...
```

Print methods are available to show relevant information. The following code prints results about risk currently stored in object `sdc`. For explanations about risk-measures, see Chap. 3 for the theory on these measures.

```
print(sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the
   data: 0
   ## Expected number of re-identifications: 24.78 (0.54  %)
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications: 117.20 (2.56 %)
   ##
   ------------------------------------------------------------------------
```

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

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

These print methods will be practically applied in Chap. 8 as soon we anonymize data sets.

More information on **sdcMicro** and its facilities can be found in the manual of **sdcMicro**, and especially in Templ et al. (2015).

1.4.4 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** (Xie 2014a).

```
args(report)

   ## function (obj, outdir = getwd(), filename = "SDC-Report", title =
   "SDC-Report",
   ## internal = FALSE, verbose = FALSE)
   ## NULL
```

shows the possible function arguments. It is obvious that the first argument should be an object of class *sdcMicroObj*. Since the report is save on the hard disk in a certain format (function argument `format`), the directory and the file name can be specified. 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.

Internal Reports:

They include information about selected key variables, performed actions, disclosure risk and data utility and session information about 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:

They contain less information than an internal report. 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.

`?report` gives more information since it gives access to the help file.

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 available because this information is not available. However, if PRAM was applied, the entire disclosure risk summary is presented differently because usual risk measures are not valid if categorical key-variables have been modified using this procedure.

We stop the discussion on reports here, but show the exact structure of a SDC report in Sect. 8 after discussing the SDC methods in the next sections and anonymizing real-world data sets.

1.5 The Point-and-Click App sdcApp

The function `sdcApp` mainly written by Bernhard Meindl see `sdcApp` envokes an easy-to-handle, highly interactive browser-based anonymization tool for users who want to use the **sdcMicro** package for statistical disclosure control but are not familiar with the native R command line interface. The software performs automated recalculation and display of frequency counts, individual and global risk measures, information loss and data utility after each anonymization step. Changes to risk and utility measurements of the original data are also conveniently displayed in the graphical user interface (GUI). Furthermore, the code of every anonymization step carried out within the GUI is saved in a script, which can easily be exported, modified and re-used, making it possible to reproduce any results.

The developed app has the following capabilities:

Link to sdcMicro: The GUI uses the functionality of the **sdcMicro** package. It allows high performance and fast computations, since all basic operations are written in computationally efficient manner.

Import/Export: Datasets exported from other statistical software, such as SAS, SPSS, Stata, can easily be imported. It is also possible to use `.csv` files as well

as data stored in R binary format. An interactive preview and selection of import parameters (such as delimiters or separators) are provided for the import of `.csv` files, which allows users to read the data correctly into the GUI. Export facilities are provided for the same formats from which data can be read into the GUI.

Usability: The app is an easy-to-use tool for anonymization of microdata, and all methods are easily accessible.

Recoding: Facilities to rename and regroup categories and change values of a variable are included.

Interactivity: Risk und utility measures are automatically estimated and displayed whenever users apply a disclosure limitation technique. In this way, users can immediately check the effects of any action. In addition, the risk and utility of the original unmodified data are displayed, which helps the user assess the effectiveness of the anonymization.

Undo: Because users can undo the last step completed in the app, they can try out several methods with different parameters and get instant feedback until the best result is achieved. Currently, users can reverse exactly one step in the history.

Reporting: Automatically generated, standardized reports in various output formats can be produced directly from the user interface. Reports can be exported to `html`, LaTeXor plain text files, see Sect. 1.4.4.

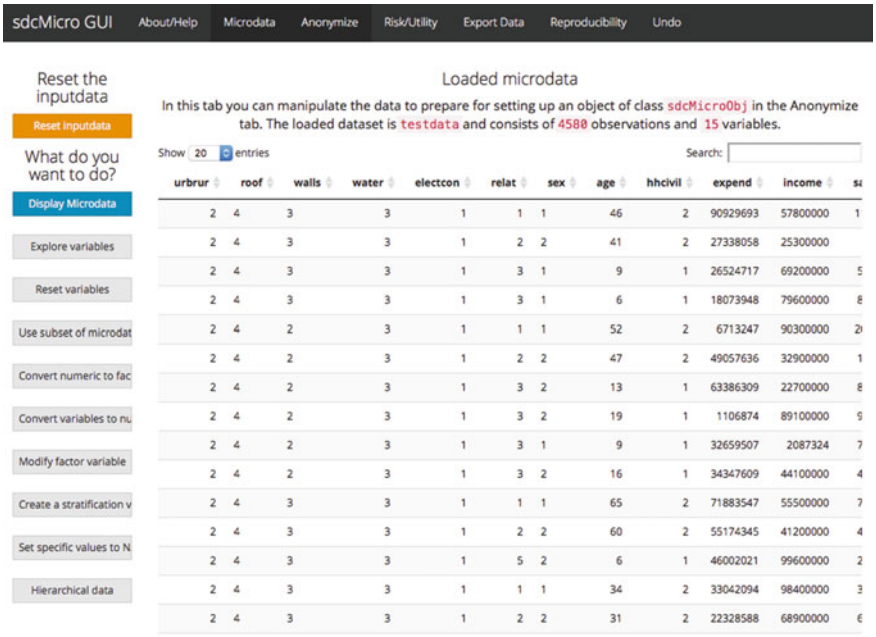


Fig. 1.3 Data manipulation view after importing a data set. Users can modify variables, generate strata variables, etc

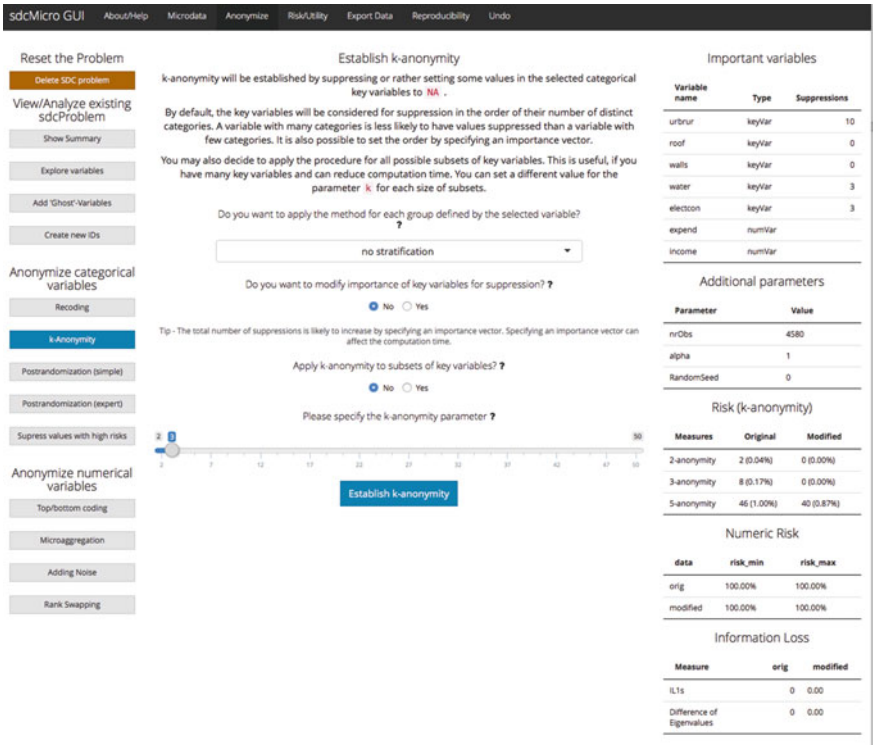


Fig. 1.4 Applying a SDC method using the app. Here, local suppression is applied to achieve 3-anonymity. Important informations are displayed on the *right*

Reproducibility: With the app, users can save, load or edit scripts for later re-use. Within the GUI, each step of the anonymization procedure is recorded and stored in a script. The script includes valid R expressions that can be copied into R . Thus, any anonymization procedure can be reproduced either by loading a script into the GUI or pasting the script directly into an R console.

The user can then interact with the interface by using standard control inputs such as buttons, drop-down menus, sliders, radio buttons or file-upload fields.

The app can be opened with

```
sdcApp()
```

In this browser window, the main sections are accessible with tabs listed in the top-navigation bar. The first step in the anonymization process is always to either upload microdata (from various formats) in tab *Microdata* or by importing a previously exported problem instance.

Once micro data have been uploaded (which are only stored locally of course), users can apply a range on methods to the currently active dataset. Figure 1.3 shows

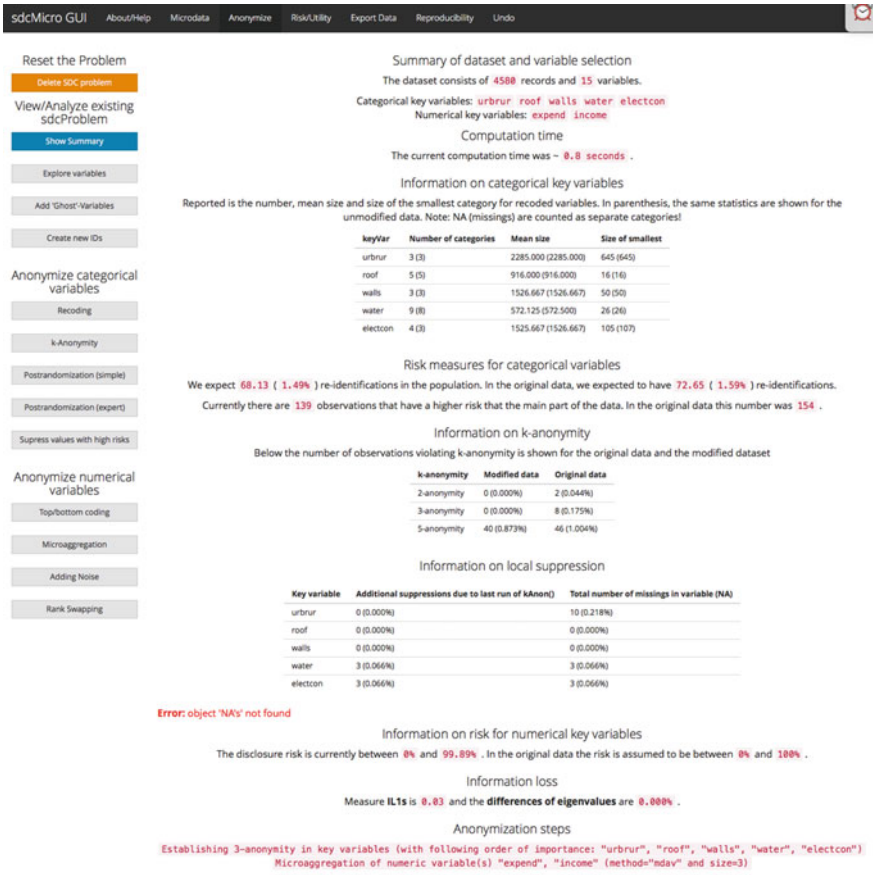


Fig. 1.5 Summary after anonymization

the data manipulation view. After microdata are ready to use, the SDC problem can be specified in the *Microdata* tab. Exemplarely, we took *urbrur*, *roof*, *walls*, *water* and *electcon* as categorical variables in the SDC problem, and variables *expend* and *income* as continuous scaled variables. Note that also sampling weights and stratification variables can be chosen as well as various other kind of variable specification can be done.

After specification of the SDC problem, a summary of the SDC problem is presented (Fig. 1.4). A bunch of SDC methods can then be applied to the SDC problem. We applied here local suppression and microaggregation and show its summary in Fig. 1.5.

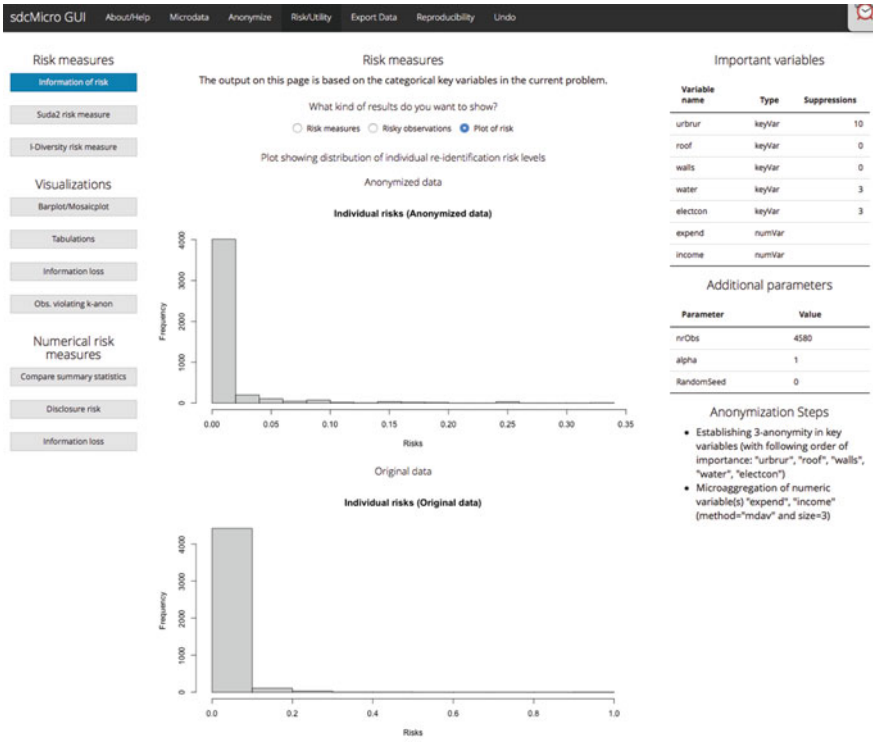


Fig. 1.6 Displaying the disclosure risk. Here, the distribution of the individual risk is compared with the individual risk of the original (unmodified) data

Various disclosure risk measures can be displayed, e.g., see Fig. 1.6 where the individual risk (Sect. 3.5) is shown using histograms.

Every action of the user in the app (click on buttons, selection from drop-down menus, etc.) is memorized and the underlying code is shown in the tab *Reproducibility*, see also Fig. 1.7. This script can be saved and used for later use, e.g. by importing it to the app.

The app `sdcApp` is not further used in the book. However, (almost) any method and code shown in the book can also be applied in the app.

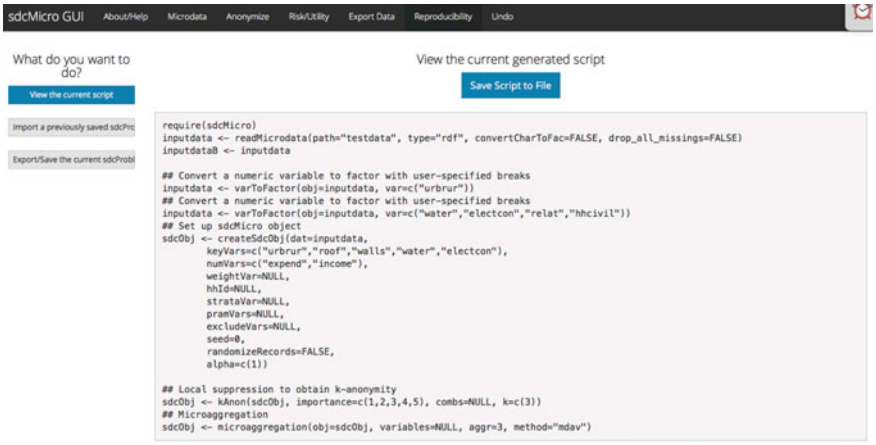


Fig. 1.7 The tab *Reproducibility* shows the code that has internally be applied to **sdcMicro** by point-and-click in the app. This code is stored for later use and reproducibility issues

1.6 The simPop package

The R package **simPop** (Templ et al. 2017) can be used to generate synthetic data. The package can be downloaded from the Comprehensive R Archive Network (CRAN). Similar to **sdcMicro**, the package supports an object-oriented S4 class structure (Chambers 2008). Parallelization is applied by default, taking into account the number of available CPUs and the structure of the data set to be generated.

The main functions are listed in Table 1.5 (see also Templ et al. 2017). All functionalities are described in the help manual, and executable examples are provided. Although most of **simPop** functions are applicable to data frames, their implementation will typically make use of objects of specific classes, in particular Templ et al. (2017):

dataObj: objects of this class contain information on the population census and survey data to be used as input for the generation of the synthetic population. They are automatically created with `specifyInput()`. They hold information on the variables containing the household and person IDs, household size, sampling weights, stratification information, and type of data (i.e., sample or a population);

simPopObj: a *simPopObj* object is created with function `createSdcObj`. Objects of this class hold information on the sample (in slot `sample`), the population (slot `pop`), optionally information on some margins in the form of a table (slot `table`), and many other information such as disclosure risk, data utility, key variables, etc. Objects in slot `sample` and `pop` must be objects of class *dataObj*. Most methods that are required to create synthetic populations can be applied directly to objects of this class.

Table 1.5 Most important functions of R-package **simPop** listed in order of a typical workflow for simulation of synthetic data

Function	Aim
`manageSimPopObj`	Accessor to objects from class *simPopObj*
`tableWt`	Weighted cross-tabulation
`calibSample`	Generalized raking procedures (iterative proportional fitting); calibrates the sample to known margins
`addWeights`	Add weights (e.g., calibrated weights) to an object of class *simPopObj*
`ipu`	Iterative proportional updating
`sampHH`	Households are drawn/sampled and new IDs are generated
`specifyInput`	Create an object of class *dataObj*
`simStructure`	Simulate the basic household structure
`simCategorical`	Simulates categorical variables
`simContinuous`	Simulates continuous variables
`simRelation`	Simulation of categorical variables
`simComponents`	Simulates components of (semi-)continuous variables
`addKnownMargins`	Add known margins (table) to objects of class *simPopObj*
`calibPop`	Calibrate a synthetic population to known margins using simulated annealing
`sampleObj`	Query or replace slot'sample' of a *simPopObj*
`popObj`	Query or replace slot'pop' of a *simPopObj*
`tableObj`	Query slot'table' of a *simPopObj*
`spCdf`, `spTable`	Weighted cumulative distribution function and cross-tabulation of expected and realized population sizes
`spCdfplot`, `spMosaic`	Plots of expected and realized population sizes

Special functionality (and not listed in Table 1.5) is available in **simPop** to apply corrections for heaping in age or income variables. In particular, the Whipple-Index (Shryock et al. 1976) and the Sprague-Index (see, e.g., Calot and Sardon 2004) are implemented with functions `whipple()` and `sprague()`, respectively. With the function `correctHeap` heaping can be corrected, whereby using truncated (log-)normal distributions. A detailed description of this functionality is out of the scope of this work, but the help pages available at `?whipple`, `?sprague` and `?correctHeap` provide details and running examples.

Applications of **simPop** are shown in the case study of Sect. 8.7. Further details on the R package is given in Templ et al. (2017).

Exercises:

Question 1.1 **Study help files**
(a) Load the **sdcMicro** package in R. Have a look at the help index.
(b) Open the help of function `createSdcObj()` and execute the examples in the example section.

Question 1.2 **Getting familiar with some data sets**
(a) Load the data set `testdata` which is available in **sdcMicro**. Look at the structure of the data using functions `str()`, `head()`, `colnames()` and `?testdata`.
(b) Load the data set `eusilc` from package **laeken**. Note that the package might need to be installed first using `install.packages("laeken")` and loaded via `library(laeken)` before you can access the data.

Question 1.3 **Getting familiar with the S4 class structure**
(a) Run the example of `?globalRecode`.
(b) List the names of all slots from the object `sdc`.
(c) Access the slot called `risk` and search for the individual risk.

References

Calot, G., & Sardon, J.P. Methodology for the calculation of Eurostat's demographic indicators. In *Population and social conditions 3/2003/F/n, Eurostat, European Demographic Observatory*, Luxembourg.

Chambers, J. M. (2008). *Software for data analysis: programming with*R. New York: Springer.

Chang, W., Cheng, J., Allaire, J. J., Xie, Y., & McPherson, J. (2016). Shiny: Web Application Framework for R.

Dai, C., Ghinita, G., Bertino, E., Byun, J.-W., & Li, N. (2009). TIAMAT: A tool for interactive analysis of microdata anonymization techniques. *PVLDB*, *2*(2), 1618–1621.

Dowle, M., Short, T., Lianoglou, A., & Saporta, R. (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. Retrieved from http://CRAN.R-project.org/package=data.table. With contributions from Srinivasan.

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

Hundepool, A., Van de Wetering, A., Ramaswamy, R., Franconi, L., Polettini, S., Capobianchi, A. et al. (2008). *μ-Argus. User Manual*, version 4.2.

Kohlmayer, F., Prasser, F., Eckert, C., Kemper, A., Kuhn, K.A. (2012). Flash: Efficient, stable and optimal k-anonymity. In *2012 International Conference on Privacy, Security, Risk and Trust (PASSAT), 2012 International Confernece on Social Computing (SocialCom)* (pp. 708–717).

Kowarik, A., Templ, M., Meindl, B., & Fonteneau, F. (2013). sdcMicroGUI: Graphical user interface for package sdcMicro, R package version 1.1.1. Retrieved from http://CRAN.R-project.org/package=sdcMicroGUI

LeFevre, K., DeWitt, D. J., & Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE 2006 (p. 25). Washington, DC, USA: IEEE Computer Society. http://dx.doi.org/10.1109/ICDE.2006.101. ISBN 0-7695-2570-9. 10.1109/ICDE.2006.101.

Nergiz, M.E., Atzori, M., & Clifton, C. (2007). Hiding the presence of individuals from shared databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD 2007* (pp. 665–676). New York, NY, USA: ACM. ISBN 978-1-59593-686-8. doi:10.1145/1247480.1247554.

R Development Core Team. (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Retrieved from http://www.R-project.org/. ISBN 3-900051-07-0.

Shryock, H. S., Stockwell, E. G., & Siegel, J. S. (1976). *The methods and materials of demography*. New York: Academic Press.

Templ, M., Meindl, B., Kowarik, A., & Dupriez, O. (2015). Simulation of synthetic complex data: The R-package simPop. *Journal of Statistical Software*, 1–38 (2017). (Accepted for publication in December 2015).

Templ, M. (2008). Statistical disclosure control for microdata using the R-package sdcMicro. *Transactions on Data Privacy*, *1*(2), 67–68. Retrieved from http://www.tdp.cat/issues/abs.a004a08.php.

Templ, M., Alfons, A., & Filzmoser, P. (2012). Exploring incomplete data using visualization techniques. *Advances in Data Analysis and Classification*, *6*(1), 29–47.

Templ, M., Meindl, B., & Kowarik, A. (2015). Statistical disclosure control for micro-data using the R package sdcMicro. *Journal of Statistical Software*, *67*(1), 1–37.

Wickham, H., & Chang, W. (2015). devtools: tools to make developing R packages easier, R package version 1.7.0. Retrieved from http://CRAN.R-project.org/package=devtools.

Xie,Y. (2014a). *knitr: A General-purpose Package for Dynamic Report Generation in R*, R package version 1.6. Retrieved from http://yihui.name/knitr/.

Chapter 2
Basic Concepts

Abstract This section introduces the basic concepts related to statistical disclosure. It presents definitions for certain groups of variables such as sensitive variables or key variables. They are crucial for any other chapter, since SDC methods differ depending on the variables chosen. In addition, basic intruder scenarios are described such as identity, attribute and inferential disclosure. The chapter ends with a discussion about the trade-off between disclosure risk and information loss. The more the disclosure risk is reduced the higher the information loss and the lower the data utility. The concept of risk-utility maps that reports this trade-off is explained based on real data.

2.1 Types of Variables

Figure 2.1 shows different kinds of variables that are important for SDC, depending on the kind of data set. For typical data sets in the bio-medical area or data from census or registers, the distinction between direct identifiers, indirect identifiers, sensitive variables and non-confidential variables is important. For surveys with complex designs, sampling weights and cluster structures are crucial as well. In the following, the different kinds of variables are explained.

2.1.1 Non-confidential Variables

For these variables, it can be assumed that no information is available in external data bases or at least that linking to external information is not possible for the non-confidential variables. In the context of SDC, they are not of great importance, since SDC methods are not applied on them. In this book, they are only considered for few aspects on data utility in Sect. 5.

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_2

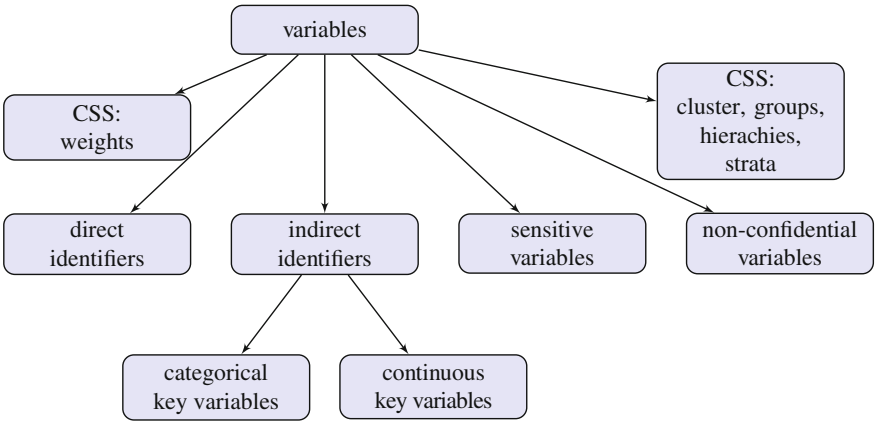


Fig. 2.1 Groups of variables and information in a data set. In addition to census, register or surveys without complex sample designs, some additional information is given for complex survey samples (CSS), e.g., the information on sampling weights, cluster structures and hierarchies in the data set. In special cases, sensitive variables can also be indirect identifiers, see Sect. 2.1.3

2.1.2 Identifying Variables

SDC methods are often applied to identifying variables whose values might lead to re-identification. Identifying variables can be further classified into direct identifiers and *key variables*.

Direct identifiers are variables that unambiguously identify statistical units, such as social insurance numbers or names and addresses of companies or persons. Removing direct identifiers is the first step of SDC.

Key variables are, in this book, defined as a set of variables that, in combination, can be linked to external information to re-identify respondents in the released dataset. Key variables are also called "indirect identifiers", "implicit identifiers" or "quasi-identifiers". For example, while on their own, the gender, age, region and occupation variables may not reveal the identity of any respondent, but in combination, they may uniquely identify respondents.

The decision on variables serving as key variables—the disclosure scenario—is always crucial and the discussion on it is continued in Sect. 2.2.1.

2.1.3 Sensitive Variables

SDC methods are also applied to sensitive variables to protect confidential information of respondents. Sensitive variables are those whose values must not be discovered of any respondent in the dataset. The determination of sensitive variables is often subject to legal and ethical concerns. For example, variables containing information

on criminal history, sexual behaviour, medical records or income are often considered sensitive. In some cases, even if identity disclosure is prevented, releasing sensitive variables can still lead to attribute disclosure (see example in Sect. 2.2.2). A variable can be both identifying and sensitive. For example, income variables can be combined with other key variables to re-identify respondents, but the variable itself also contains sensitive information that should be kept confidential. On the other hand, some variables, such as occupation, might not be sensitive but could be used to re-identify respondents when combined with other variables. In this case, occupation is a key variable and SDC methods should be applied to it to prevent identity disclosure.

2.1.4 Linked Variables

Often, it is important that the cell value of the same observation of another ("linked") variable should automatically be suppressed as well when a cell is suppressed in a particular key variable. We also use the term *ghost variables* when such variables are linked to some categorical key variables. After applying local suppression the ghost variables should have the same suppression pattern as to those variables to that they are linked. A practical example of ghost variables and how to deal with them is given in Chap. 4, Sect. 4.2.2.7.

2.1.5 Sampling Weights

Sampling has an effect on the disclosure risk and are taken into account for estimation of the disclosure risk (see Chap. 3). It is obvious that the risk is higher for a whole target population (e.g. census) than for sample data. Sampling introduces additional uncertainty about whether the unit identified in statistics is the particular unit in the population, since it is usually not known if the particular respondent participated in the survey. However, if a unit is included in the sample and this unit is unique in the population, this unit might have the same risk, as if the sample were equal to the whole population, i.e. the same risk like as the sampling weight is 1.

Data collected based on complex survey designs typically include at least one vector on sampling weights since each individual may have an unequal chance of being selected. For instance, single-parent households will be given a greater chance of being selected in a survey with the aim of estimating low incomes, and weights can adjust for this. Moreover, weights can also serve other purposes, such as calibration of survey samples to exactly fit known population frequencies and helping to correct for non-response.

In addition, the weights themselves can lead to identifying units. For example, in stratified simple random sampling designs, the weights might be the same for each individual in the strata. If the strata variable is a key variable that is anonymized, the sampling weights may make this anonymization useless because we know, based on the sample weights and some knowledge on the sampling design, which unit is in the same strata.

In summary, the sampling weights have to be considered for disclosure risk estimation (see Chap. 3) and in addition, if they report confidential grouping information, they also might be anonymized by SDC methods for continuous data (see Chap. 4).

2.1.6 Hierarchies, Clusters and Strata

Data may include hierarchies or clusters. This is often also related to sampling designs. In the Austrian Structural Earning Statistics (SES), for example, enterprises are drawn in the first stage and employees are drawn from the selected enterprises in the second stage. In the European Statistics on Income and Living Conditions (EU-SILC), for every member of a household information is retrieved (cluster design). The estimation of disclosure risk is then different from the one without having such clusters. For more details, see Sect. 3.6.

A SDC anonymization method may be applied independently on subgroups (defined by stratification variables). For example, if a grouping procedure such as microaggregation (see Sect. 4.3.1) is applied to economic data, a steel producer's continuous values might be aggregated with a enterprise that has its production in agriculture. However, for data utility aspects, it is often better to apply microaggregation in each economic branch independently.

2.1.7 Categorical Versus Continuous Variables

SDC methods differ between categorical variables and continuous variables. A categorical variable takes values over a finite set. For example, gender is a categorical variable. A continuous variable is numerical; arithmetic operations for real numbers can be performed with it. For example, personal income and turnover of enterprises are continuous variables.

2.2 Types of Disclosure

Three types of disclosure are noted here (see also, Lambert 1993).

Suppose a hypothetical intruder has access to some released microdata and attempts to identify or find out more information about a particular respondent.

Disclosure, also known as "re-identification", occurs when the intruder reveals previously unknown information about a respondent by using the released data.

2.2.1 Identity Disclosure

Identity disclosure occurs when the intruder successfully associates/links a known individual with released data. For example, the intruder links a released data observation with external information or identifies a respondent with extreme data values. In this case, an intruder can exploit a small subset of variables to make the linkage and, once the linkage is successful, the intruder has access to all other information in the released data related to the specific respondent.

Even direct identifiers such as names, addresses or social security numbers are deleted from the data set, still extreme values will include high risk of disclosure. To give an extreme example, *occupation* = *president of USA* will surely lead to a disclosure. Also an observation with a value on income of 1.000.000.000€ is of high risk because there will not be many people among the population with such an income. Even more, rare attribute combinations of indirect identifying variables available in public databases are potential of high risk. For example, if an individual having the unique combination of attributes *state* = *AT*, *ethnicity* = *Korean*, *age* = 50, *gender* = *female* and *occupation* = *university lecturer* is in the release sample, this person most probably is also unique in the population and therefore of high risk of disclosure. As soon as the individual is (re-)identified, the intruder knows the possible sensitive values of any variables of this person.

2.2.2 Attribute Disclosure

Attribute disclosure occurs when the intruder is able to determine some new characteristics of an individual based on the information available in the released data. Membership disclosure can be seen as a specific type of attribute disclosure and for disclosure of microdata this is the most relevant case of the more general definition of attribute disclosure.

Table 2.1 serves as an example of (membership) attribute disclosure. In this toy data set, three categorical key variables are defined and the variable *religion* serves as sensitive variable. All in all, four observations are present in this data set. Identity disclosure is not possible since all persons have the same entries. However, we immediately know that each person in this sample with the combination *race* = *black*, aged 50–60, living in region *ZIP* = 1234 is *roman/catholic*. This leads to a disclosure for the sensitive variable "*religious view*".

Table 2.1 Small example showing a case of successful attribute disclosure

OBS	Key variables			Sensitive variable
	Race	Age	Region	Religion
1	Black	50–60	1234	Roman/catholic
2	Black	50–60	1234	Roman/catholic
3	Black	50–60	1234	Roman/catholic
4	Black	50–60	1234	Roman/catholic

Another example: If a hospital publishes data showing that all female patients aged 56 to 60 have cancer, an intruder then knows the medical condition (=cancer) of any female patient aged 56 to 60 staying in this hospital without having to identify the specific individual.

2.2.3 Inferential Disclosure

Inferential disclosure occurs when the intruder is able to determine the value of some sensitive characteristic of an individual more accurately with the released data than it would have been possible otherwise. Inferential disclosure happens when individual's sensitive characteristics can be well predicted from a good model applied on the released data.

For example, with a highly predictive regression model, an intruder will be able to infer a respondents sensitive income information using attributes recorded in the data, leading to inferential disclosure.

In practice, a model would be fit onto the released data and external information on an individual would be used to predict attributes of this individual. For example, assume that *age*, *gender*, *region*, *economic status*, *economic status* and *income* are available in released data. The intruder also has information on the first five mentioned variables on a particular person, say *A*. He can fit a regression model with income as response on the released data and he will receive the fitted regression coefficients. A linear combination of these coefficients with the values on person A predicts the income of this individual. Depending on the quality of the model, the income might be fitted accurately enough. If this is the case, the intruder successfully disclosed the income of person A.

As an example, we use the EU-SILC data set from R package **laeken** (Alfons and Templ, 2013). This data set is synthetically generated. However, to show inferential disclosure, we assume that the values are real except the income components. Let's also assume that an intruder is interested in the variable employee cash or near cash net income (variable `py010n`) of a person with the following attitudes:

```
intrudersKnowledge <- data.frame("hsize" = 3,
                                 "db040" = "Tyrol",
                                 "age"= 34,
                                 "pl030" = "2",
                                 "pb220a" = "AT",
                                 "eqIncome" = 16090)
intrudersKnowledge

  ##   hsize db040 age pl030 pb220a eqIncome
  ## 1     3 Tyrol  34     2     AT    16090
```

Note that this is already more information than typically available, since the intruder even knows the equivalized income of persons in the household.

He is interested in knowing the employee cash or near cash net income of a person. The intruder might try to find a good model using personal income on employee cash or near cash net income as response and certain other variables as predictors. For simplicity, assume that he is just using his predictors without any interaction terms (the inclusion of interaction terms do not improve the predictive power for this data set anyhow). Since the employee cash net income variable is right-skewed, the log is taken. Moreover, assume the intruder knows that the income is not 0, so we only take observations with cash net incomes greater than zero.

```
data(eusilc)
mod1 <- lm(log(py010n) ~ hsize + db040 + age +
                         pl030 + pb220a + eqIncome,
           data=eusilc[eusilc[,"py010n"] > 0, ])
s1 <- summary(mod1)
s1$r.squared

  ## [1] 0.3682565
```

We see that the predictive power of the model is not very high ($R^2 \sim 0.37$).

Let us assume that the real (unknown!) value of `py010n` for person 1 is 9500€. We predict this value from our given data set and estimated model.

```
exp(predict(mod1, intrudersKnowledge))

  ##        1
  ## 7242.003
```

We get an estimated value of 7.242003×10^3 on employee cash or near cash net income. We can ask ourselves if this value is far enough from the true unknown value of 9500€. However, we should also take the model uncertainty into account.

```
exp(predict(mod1, intrudersKnowledge, interval = "prediction"))

  ##        fit      lwr      upr
  ## 1 7242.003 1826.026 28721.73
```

We see that the prediction interval is rather large, inferential disclosure is hardly possible with this scenario.

Exercises:

Question 2.1 **Choice of variables (I)**
Have a brief look at a popular data set, the EU-SILC data in R-package **laeken** (Alfons and Templ 2013). Read the help for this data set by typing in R:

```
install.packages("laeken")
data(eusilc)
?eusilc
```

Determine which of the variables should be defined as
(a) direct identifiers (if any)
(b) categorical key variables
(c) continuous key variables
(d) sensitive variables

Question 2.2 **Choice of variables (II)**
Please have a brief look at another popular data set, the Structural Earnings Statistics in R-package **laeken** (Alfons and Templ 2013). Read the help for this data set by typing in R:

```
data(ses)
?ses
```

Determine which of the variables should be defined as
(a) direct identifiers (if any)
(b) categorical key variables
(c) continuous key variables
(d) sensitive variables

Question 2.3 **Frequencies of key variables**
Use again the EU-SILC data set from package **laeken**. Is a re-identification of observation 8 possible assuming region (*db040*), age, gender (*rb090*) and economic status (*pl030*) as categorical key variables. Is re-identification of observation 3 easily possible?

If you do not have any experience in R, please read Chap. 1 first, before starting this exercise.

2.3 Disclosure Risk Versus Information Loss and Data Utility

Applying SDC techniques to the original microdata will result in information loss and hence affect data utility. Data utility describes the value of data as an analytical resource, comprising analytical completeness and analytical validity.

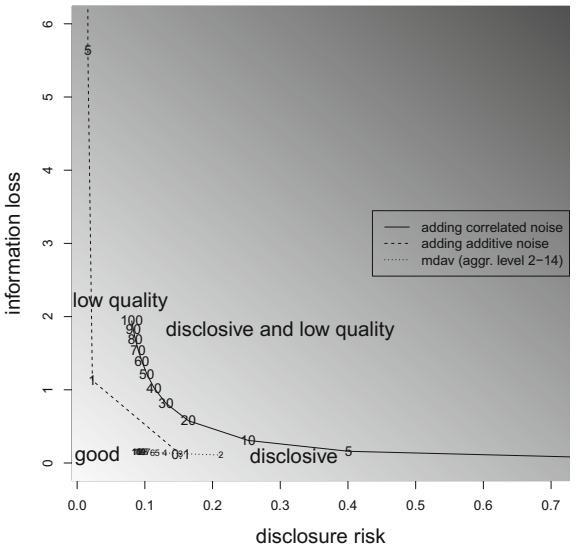


Fig. 2.2 Disclosure risk versus information loss obtained from two specific SDC methods applied to the SES data. Note that the information loss for the original data is 0 and the disclosure risk is 1 respectively, i.e. the curves theoretically start from (1, 0)

Therefore, the main challenge for a statistical agency is to apply the optimal SDC techniques that reduce disclosure risks with minimal information loss, preserving data utility. To illustrate the trade-off between disclosure risk and information loss, Fig. 2.2 shows a general example of results after applying two different SDC methods to the Structure of Earnings Statistics (SES) data (cf. Chap. 8). Please note that the specific SDC methods and measures of disclosure risk and information loss will be explained in the following sections.

Before applying any SDC methods, the original data is assumed to have information loss of 0. As shown in Fig. 2.2, three different SDC methods are applied to the same dataset with varying parameters. The solid curve represents the first SDC method (i.e., adding correlated noise; see Sect. 4.3.2). The curve illustrates that, as more noise is added to the original data, the disclosure risk decreases but the extent of information loss increases. This is also visible with method 2 (adding additive noise; see Sect. 4.3.2), but it is more extreme. Small noise is enough to decrease the quality of the data. In comparison, the dotted curve, illustrating the result of the third SDC method (micro-aggregation; see Sect. 4.3.1), is much less steep than the solid and dashed curves representing the first and second method. In other words, at a given level of disclosure risk (for example, when disclosure risk is 0.1) the information loss resulting from the second method is much lower than that resulting from the first and second method. Therefore, for this specific dataset and the tested methods, method 3 with parameter greater than 5 is the preferred SDC method for the statistical agency to reduce disclosure risk with high data utility.

To check if the anonymized data has a similar structure as the original data, certain metrics can be used as information loss measures. For example, the number of categories of a variable in the original data set can be compared with the number of categories in the anonymized data set. We come back to this issue in Chap. 5.

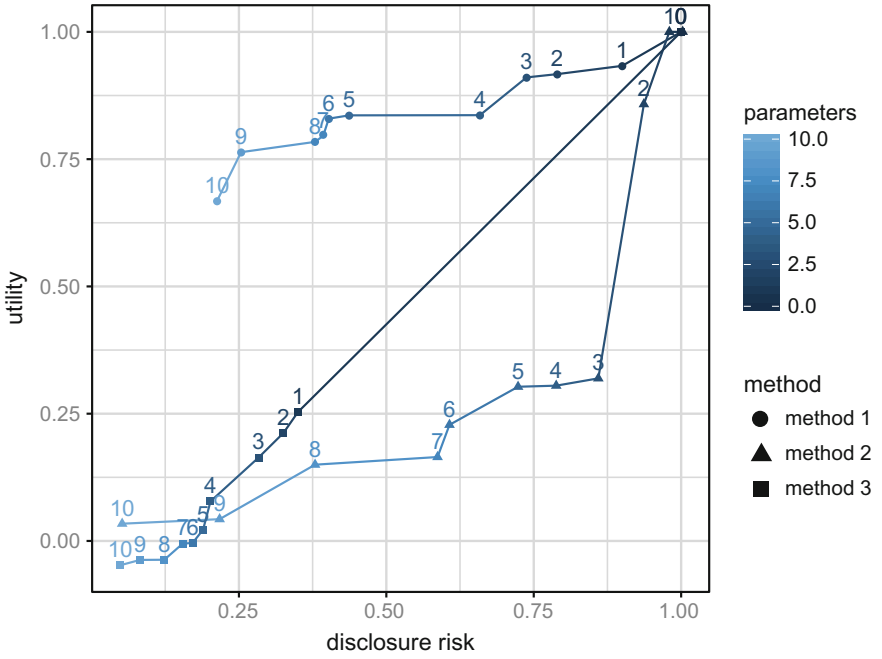


Fig. 2.3 Risk-utility map for three methods

Often not the information loss but the data utility is estimated. For example, a risk-utility map typically includes the data utility instead of the information loss. Theoretically, its name should then be a risk-information-loss map, however, we do not change the term risk-utility map in the following. The data utility can be estimated, for example, on results of regression analysis, i.e. to compare regression coefficients obtained from original and anonymized data. A typical risk-utility map looks like Fig. 2.3. The higher the parameter for perturbation, the lower the disclosure risk but also the lower the data utility. It is clearly visible that method 1 outperforms the other two methods, while method 2 is still better than method 3. For example, with parameter 7, the utility is still high but the disclosure risk reduced significantly. However, the data holder still has to decide if the loss in data utility is still too high while the disclosure risk did not reduce to a given specified threshold. Method 2 is not applicable for this data set. If the parameter for perturbation is low, the risk is too high. If the parameter value increases, the data utility is unacceptably low. Almost the same picture describes method 3.

However, if the utility measure is changed, or the disclosure risk method, the picture would change. In any case, the trade-off between risk and utility always exists and a compromise between disclosure risk and utility always has to be made. The lower the disclosure risk the lower the data utility.

More theory on data utility and information loss can be found in Chap. 5 and practical applications are included in Chap. 8.

However, not only the choice of the information loss criteria or data utility measure is crucial, but also the choice of the risk measure is important. This is discussed in Chap. 3.

Question 2.4 **Risk-utility maps**

Have a look at Fig. 2.2. Assume that the lawyers at your organisation determine the maximum tolerable risk of 0.07. In this case, less than 7% of the observations can be matched correctly. Please answer the following questions.

(a) Which method do you choose—the method corresponding to the lower curve (microaggregation), the method corresponding to the dashed line (adding additive noise) or the method represented by the upper solid-line curve (adding correlated noise)?
(b) The threshold on risk is determined at 0.09 and maximum information loss at 0.5. Your company wants to use adding additive noise to continuous key variables. Would you agree to choose this method?
(c) The risk threshold is 0.2 and the maximum information loss should be 1.5. Which method would you choose?

2.4 Release Types

Confidentiality aspects and the accepted level of disclosure risk depends on national laws, on the type of users and on the type of release of data.

Generally, data release methods can be classified in five different types.

2.4.1 Public Use Files (PUF)

In simple words, this type of data is accessible by the public mostly without any conditions, sometimes with easy-to-meet conditions, e.g. by registration with email, name and address details.

The quality of PUFs varies depending on the needs of the users. Sometimes a lot of effort is spent to produce a PUF that is close-to-reality but at low disclosure risk. Such data sets are then useful for (mostly) researchers to develop methods. But such PUFs are also very useful for teaching. However, sometimes the PUFs are produced with low quality because they are only needed by researchers to make their computer code run on the data set (see Sect. 2.4.4).

In general, PUFs are made easily accessible to

- let researchers develop methods on close-to-reality data;
- for remote execution tasks (Sect. 2.4.4);
- lecturers to support teaching with close-to-reality data sets.

The risk of identifying individual respondents in a PUF should converge to zero. Minimizing the risk of disclosure involves eliminating direct identifiers and modification of indirect identifiers (see Sect. 2.1.2), e.g. by recoding and local suppression (see Sect. 4.2). Also, outliers in continuous scaled variables might be removed and continuous variables might be perturbed (see Sect. 4.2). Other very promising methods include the simulation of synthetic data Alfons et al. (2011), Templ and Filzmoser (2014), Drechsler et al. (2008), Templ et al. (2017) (see Chap. 6 and also Sect. 8.7). Such methods should incorporate sampling designs, missing values, hierarchical and cluster structures (such as persons in households or employees in enterprises). Generally, close-to-reality data sets can be simulated synthetically including very low disclosure risk (Templ and Alfons 2010).

2.4.2 Scientific Use Files (SUF)

SUFs are microdata whereby the researchers need a licence or contract to access them, however, often only researchers can get a contract. In any case, the users of SUFs need authorization to access such data sets.

SUFs are commonly less restrictive than PUFs according to the disclosure risk, but the risk of disclusure still should be low. Of course, direct identifiers are removed as well and anonymization is applied to indirect identifiers. For a data provider, it is recommended that the potential users are asked to complete an application form to demonstrate the need to use a licensed file for a stated statistical or research purpose (Dupriez and Boyko 2010). This allows the data producer to learn which characteristics and data analysis are important for the users. This may also lead to an adaptation of the anonymization methods applied to optimize user needs.

2.4.3 Controlled Research Data Center

Microdata available in a controlled research data center are usually provided at computers at the site of the data provider and are not linked to the internet. Usually, no information can be downloaded via USB ports, CD-DVD or any other device. The accessible data sets may include high risk of disclosure although direct identifiers are removed. Users may only have access to one data set for which they also signed an agreement for use, have to specify why they need access and they typically have to report the goal of their research. The final output is checked by the staff of the data provider (usually experts in statistical disclosure control) and only output with very low risk of disclosure is finally given to the user for external use, e.g. in researchers publications. In any case, the costs for running a research data center might be high because of the staff and facilities needed.

2.4.4 Remote Execution

With remote execution researchers, send their code to the data provider and the data provider executes the code on the original files, checks the output and sends the output that does not disclose information back to the researchers. This is usually an iterative process. The researcher will check their results, and depending on the previous results, they might modify their code or write a more detailed code for analysis.

The costs of remote execution might be reduced by providing structural files. Such files have the same levels of variables as the original data sets, but simulated in a very simplified manner, often just by sampling variable-wise. The structure of structural files might differ a lot from the original data sets, but researchers can check if their code runs. As an alternative to structural files, close-to-reality synthetic PUFs might be provided so that the researchers can test their code before submitting the code to the data provider.

2.4.5 Remote Access

Remote access is the NSI's dream. It would only be necessary to have the data on a server, make a secure connection, install necessary software and finally check the output from the researcher once.

The researcher uses their own desktop computer to connect with a secure connection to the server of the data providing agency. The researcher then usually has access to raw unmodified microdata, but it is not possible to download the data or generated results. The researcher, however, can look at the data and work with it depending on the software installed on the server. Using the pre-installed software, the researcher can manipulate microdata without any restriction. The final results are checked by the data holder and those results which fulfil all criteria for confidentiality are sent to the researcher. Of course, additional attempts may contain the logging of the work of the researchers.

However, this dream hardly ever comes true because of legislative restrictions and practical limitations. For example, even displaying the unmodified raw microdata violates the Austrian law on data privacy. Thus, the researcher may only get access to modified microdata (scientific-use files). In a remote access environment, it is possible to report query results from the underlying original data, but certain queries can also disclose confidential information. It is also out-of-scope that a NSI will check every result that a researcher wants to publish against confidentiality, since the methods (out of thousands of available methods) may vary a lot. Also, it often cannot be precisely determined if a result is dislosive or not. Moreover, queries that disclose information can be programmed in such a manner that there is no chance to detect if such a query discloses information. In any case, strict penalties for any misuse of the data should be executed. Another practical problem is that only pre-defined

software products are available on the server, but in nowadays world researchers may need a great number of individual software packages that may depend on many other software products. Maintaining the server is thus time consuming. Beside all the mentioned risks, various countries offer (successful) remote access facilities and the users usually also have to sign a contract against misuse of the provided remote access system.

References

Templ, M., & Alfons, A. (2010). Disclosure risk of synthetic population data with application in the case of EU-SILC. In *Privacy in Statistical Databases*. Lecture Notes in Computer Science, pp. 174–186. Springer. ISBN 978-3-642-15837-7.

Templ, M., Meindl, B., Kowarik, A., & Dupriez, O. (2017). Simulation of synthetic complex data: the R-package simPop. *Journal of Statistical Software*, 1–38. Accepted for publication in December 2015.

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.

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 and Applications*, 20(3), 383–407. http://dx.doi.org/10.1007/s10260-011-0163-2.

Drechsler, J., Bender, S., & Rässler, S. (2008). Comparing fully and partially synthetic datasets for statistical disclosure control in the German IAB Establishment Panel. *Transactions on Data Privacy*, *1*(3), 105–130.

Dupriez, O., & Boyko E. (2010). Dissemination of microdata files. Formulating policies and procedures. IHSNWorking Paper No 005, Paris: International Household Survey Network.

Lambert, D. (1993). Measures of disclosure risk and harm. *Journal of Official Statistics*, *9*, 313–331.

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.

Chapter 3
Disclosure Risk

Abstract One of the key tasks in SDC is to estimate the disclosure risk of individuals but also to estimate a global risk for the whole data set. A very basic idea is to calculate frequency counts of the categorical key variables. The concept of uniqueness and the concept of k-anonymity and l-diversity are important and outlined first. SUDA is extending the concept of k-anonymity it also searches for uniqueness in subsets of key variables. For surveys from complex designs, the estimation of frequency counts in the population and sample is of central interest. Mainly two approaches are used: the individual risk approach and the estimation of the global risk by log-linear models. For continuous key variables, other concepts are used to estimate the disclosure risk. They are rather based on distances than on counts. The risk estimation concepts presented here evaluate original data sets or data sets that are modified through traditional (perturbative) anonymization methods.

3.1 Introduction

Disclosure risk is defined based on assumptions of disclosure scenarios, that is, how the intruder might exploit the released data to reveal information about a respondent. For example, an intruder might achieve this by linking the released file with another data source that shares the same respondents and identifying variables. In another scenario, if an intruder knows that his/her acquaintance participated in the survey, he/she may be able to match his/her personal knowledge with the released data to learn new information about the acquaintance. In practice, most of the measures for assessing disclosure risks, as introduced below, are based on key variables, which are determined according to assumed disclosure scenarios. Risk assessment measures also differ for categorical and continuous variables.

A considerable number of research has been carried out in the area of statistical disclosure risk estimation. Some of the most influential works are Carlson (2002a, b), Hundepool et al. (2012), Willenborg and De Waal (2000), Skinner and Shlomo (2006).

Disclosure risk arises if a given data set is released. It is assumed that the risk r takes a non-negative real value ($r \geq 0$) and a risk of zero ($r = 0$) indicates no risk. Measuring the disclosure risk in a microdata set is a key task. Risk measures are

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_3

essential to be able to decide, if the data set is protected enough to be released. If the data set is not protected enough certain anonymization methods may reduce the disclosure risk (for anonymization methods, see Chap. 4).

In general, disclosure risk methods differ between categorical and continuous key variables. In the first subsections of this chapter only methods for categorical key variables are discussed. Thus a subset $\mathbf{Z}$ from the data set $\mathbf{U}$ is considered, with $\mathbf{d_j} \in \mathbf{Z}$ and $\mathbf{d_j}$ determines the j-th categorical variable.

Before estimating the disclosure risk, a closer look at frequency counts is necessary.

3.2 Frequency Counts

Computing frequency counts serves as a basis for many disclosure risk estimation methods. Therefore, this topic is intensively discussed on the next pages.

Consider a sample $\mathbf{S}$ with n observations or/and a finite population $\mathbf{U}$ with N observations. In general, if the frequency counts are estimated/calculated from a finite population $\mathbf{U}$ of size N, we speak about *population frequency counts*, if they are calculated from the sample $\mathbf{S}$, the term *sample frequency counts* is used.

The frequency counts can be computed for a combination of q variables that gives the distribution of frequency counts. The variables $\mathbf{Z}_1, ..., \mathbf{Z}_q$ have to be categorical, with $C_1, ..., C_q$ characteristics respectively, i.e. $C_j = |\mathbf{Z}_j|$ is the amount of categories from one variable.

Cross tabulation is a statistical process that summarizes categorical data to create a *contingency table*. A contingency table itself is a type of table that displays the frequency distribution of the categorical variables. The elements of a contingency table are denoted as cells. Every cell shows the frequency of one key whereas a key is one combination of categorical key variables.

All combinations of categories in the key variables can be calculated by cross tabulation of these variables. Each combination of values (=keys) defines a cell in the table. The maximum number of all possible cells is given by $\prod_{i=1}^{q} C_i = C$.

Let $\mathbf{X}$ be the table of all combinations, which is for simplicity labeled as $1, 2, ..., C$. The different categories C of $\mathbf{X}$ divide the population $\mathbf{U}$ or the sample $\mathbf{S}$ into C subpopulations/subsamples $\mathbf{U}_j \subseteq \mathbf{U}$ or $\mathbf{S}_j \subseteq \mathbf{S}$ respectively, with $j \in \{1, ..., C\}$.

To give an another example for keys, we consider two categorical key variables $\mathbf{Z}_1$ (gender) and $\mathbf{Z}_2$ (eye-color) given, with $C_1 = |\mathbf{Z}_1| = 2$ (“man”, “woman”) and $C_2 = |\mathbf{Z}_2| = 3$ (“blue”, “brown”, “green”) characteristics. Then there exist 6 keys, e.g. (“man”, “blue”) or (“woman”, “green”).

The subpopulation $\mathbf{U}_j \subseteq \mathbf{U}$ or subsample $\mathbf{S}_j \subseteq \mathbf{S}$ contains all observations belonging to the j-th key, with $j \in \{1, 2, ..., C\}$. To give an example: if there are exactly five observations in a subpopulation $\mathbf{U}_j$ with the key: *woman*, *student*, *blue*. This key yields subpopulation $\mathbf{U}_{woman,student,blue} \subseteq \mathbf{U}$ with $|\mathbf{U}_{woman,student,blue}| = 5$.

The population frequency counts F_j with $j \in \{1, ..., C\}$ are the numbers of observations belonging to subpopulation $\mathbf{U}_j$, i.e. $F_j = |\mathbf{U}_j|$. Consider a random sample $\mathbf{S} \subseteq \mathbf{U}$ of size $n \leq N$ drawn from a finite population $\mathbf{U}$ of size N. Let π_j with $j \in \{1, 2, ..., N\}$ be the inclusion probabilities, which is the probability that a record $\mathbf{x}_j \in \mathbf{U}$ is chosen in the sample. The sample frequency counts are analogously defined as the population frequency counts F_j and denoted by f_j. Thus the sample frequency counts f_j with $j \in \{1, ..., C\}$ are the numbers of records belonging to subsample $\mathbf{S}_j$, i.e. $f_j = |\mathbf{S}_j|$.

In R the function `table()` can be used to compute contingency tables on sample level. When having a population, this function can also be applied on population level. Otherwise, to estimate population frequencies, the function `tableWt()` is useful. It takes sample weights into account for Horwitz-Thompson weighted estimates (Horvitz and Thompson 1952) of population frequencies. Exemplary, the sample frequency calculation is shown based on two key variables for the Austrian EU-SILC data.

```
## sample frequency counts
table(eusilc[,c("db040", "pb220a")])
```

```
##                pb220a
## db040             AT   EU Other
##   Burgenland     453   16     7
##   Carinthia      842   19    26
##   Lower Austria 2207   26   107
##   Salzburg       655   25    83
##   Styria        1783   24    73
##   Tyrol          928   43    50
##   Upper Austria 2056   45   143
##   Vienna        1641   76   221
##   Vorarlberg     508    9    41
```

3.2.1 The Number of Cells of Equal Size

The number of cells of equal size (cell sizes) are obtained as follows. T_j is the number of cells of size j, i.e.

$$T_j = \sum_{i=1}^{C} \mathbb{1}(F_i = j), \; j = 0, 1, ..., N \quad , \tag{3.1}$$

The sample counterpart t_j is given by

$$t_j = \sum_{i=1}^{C} \mathbb{1}(f_i = j),\ j = 0, 1, ..., n \quad , \tag{3.2}$$

where $\mathbb{1}_{\mathbf{A}}$ denotes the characteristic function of a subset $\mathbf{A}$ of a set $\mathbf{X}$, with $\mathbb{1}_{\mathbf{A}} : \mathbf{X} \rightarrow \{0, 1\}$ and

$$\mathbb{1}_{\mathbf{A}}(\mathbf{x}) := \begin{cases} 1 & \text{if } \mathbf{x} \in \mathbf{A} \\ 0 & \text{if } \mathbf{x} \notin \mathbf{A} \end{cases} \quad .$$

The above definitions of T_j and t_j with $j \in 1, 2, ...C$ determines cell size indices of the population and sample. It is clear that there is a relation between T_j and F_i as well as for t_j and f_i. The relation between T_j and F_i

$$\sum_{j=1}^{N} jT_j = N = \sum_{i=1}^{C} F_i$$
$$\sum_{j=1}^{n} jt_j = n = \sum_{i=1}^{C} f_i \quad ,$$

we can proof as follows.

Proof

$$\bigcup_{i=1}^{C} \mathbf{U}_i = \mathbf{U} \; and \; \mathbf{U}_i \cap \mathbf{U}_j = \emptyset,\ \forall i \neq j$$
$$\Longleftrightarrow |\bigcup_{i=1}^{C} \mathbf{U}_i| = |\mathbf{U}|$$
$$\Longleftrightarrow \bigcup_{i=1}^{C} |\mathbf{U}_i| = |\mathbf{U}|$$
$$\Longleftrightarrow \sum_{i=1}^{C} F_i = N$$

□

As mentioned above, there exists also a relation between the number of combinations and the cell size indices T_i and t_i:

$$\sum_{j=0}^{N} T_j = \sum_{j=0}^{n} t_j = C \quad .$$

Proof of aboves equation, see also Sect. 3.2.1.

$$\begin{aligned}
\sum_{j=0}^{N} T_j &= \sum_{j=0}^{N} \sum_{i=1}^{C} \mathbb{1}(F_i = j) \\
&= \sum_{i=1}^{C} \sum_{j=0}^{N} \mathbb{1}(F_i = j) \overset{(1)}{=} \sum_{i=1}^{C} 1 = C \\
\sum_{j=0}^{n} t_j &= \sum_{j=0}^{n} \sum_{i=1}^{C} \mathbb{1}(f_i = j) \\
&= \sum_{i=1}^{C} \sum_{j=0}^{n} \mathbb{1}(f_i = j) \overset{(1)}{=} \sum_{i=1}^{C} 1 = C
\end{aligned}$$

(1) *because* $0 \leq F_i \leq N$ *and* $0 \leq f_i \leq N, \ \forall i \in 1, ..., C.$ □

3.2.2 Frequency Counts with Missing Values

The following R code shows the frequency counts calculation with three categorical key variables (federal state (`db040`), household size (`hsize`) and citizenship (`pb220a`)) from the data set `eusilc`. The package **data.table** is used for fast calculations. For each key the frequencies are calculated with the following code.

```
dt <- data.table(eusilc)
dt[, .N ,by = list(db040, hsize, pb220a)]

##           db040 hsize pb220a   N
##   1:      Tyrol     3     AT 179
##   2:      Tyrol     3  Other   4
##   3:      Tyrol     3     NA  43
##   4:      Tyrol     4     AT 247
##   5:      Tyrol     4     NA 140
##  ---
## 221: Vorarlberg     2  Other   1
## 222:      Tyrol     8     AT   1
## 223:      Tyrol     7     AT   4
## 224:      Tyrol     7     NA   3
## 225: Burgenland     1     EU   1
```

There exist 225 combinations/keys. However, we see that missing values (NA) are considered as own category. However, given a missing value, an intruder must not know the true category. In the worst case scenario, he can predict the category correctly. In the best case scenario, the probability of guessing the correct category is one divided by the number of categories. In SDC, usually this second assumption is taken into account (see, e.g., Franconi and Polettini 2004).

However, more scenarios are thinkable. In the following five scenarios are reported how sample frequency counts can be calculated.

1. (current method) Missing values increase frequencies in other categories.
2. (conservative method) Misssing values do not increase frequencies in other categories but in those observation where a missing occurs.
3. (category size) Missing values do increase frequencies in other categories by a factor c. This method can be used as a general method to account for missing values in frequency calculations.
4. (conservative method 2) Misssing values do not increase frequencies in other categories.
5. (own category) Same as method 4, but missings are treated like an own category.

A (very) simple table should illustrate these methods. The default method in **sdcMicro** is based on the assumption that a missing value can stand for any category and so a missing value in a variable can increase the frequencies of several keys. For example, the frequency counts of the left table would lead to frequencies represented in the middle table (default approach) or in the right table (conservative approach).

key1	key2	key3
1	1	3
1	1	NA
2	1	3
NA	1	NA

$\longrightarrow$

key1	key2	key3	f_k
1	1	3	3
1	1	NA	3
2	1	3	2
NA	1	NA	4

till

key1	key2	key3	f_k
1	1	3	1
1	1	NA	3
2	1	3	1
NA	1	NA	4

The discussion on missing values continues in the next section, but also it is a topic in Sect. 4.2.2, because it makes sense to discuss it together with the topics related to local suppression.

3.2.3 *Sample Frequencies in sdcMicro*

The `sdcMicro` package provides the function `freqCalc` or `measure_risk` which can also be used to compute the (sample) frequency counts.

Frequency counts are automatically estimated when creating a *sdcMicroObj* object. The general extractor function `get.sdcMicroObj` can be used (`slot` is equivalent) to extract sample and population frequencies from the current object:

```
sdc <- createSdcObj(eusilc,
          keyVars = c("db040", "hsize", "pb220a"),
          weightVar = "rb050", hhId = "db030")
head(get.sdcMicroObj(sdc, type="risk")$individual)

   ##               risk  fk        Fk    hier_risk
   ## [1,] 8.967734e-06 222 112014.46 6.044731e-05
   ## [2,] 4.308265e-05  47  23714.77 6.044731e-05
   ## [3,] 8.397756e-06 237 119583.00 6.044731e-05
   ## [4,] 5.250816e-06 387 190938.97 2.046126e-05
   ## [5,] 5.250816e-06 387 190938.97 2.046126e-05
   ## [6,] 4.979891e-06 408 201300.00 2.046126e-05
```

The column denoted by `fk` includes the sample frequency counts assigned to each observation, the other columns are discussed later.

But also without creating an object of class `sdcMicroObj`, one can use the function `freqCalc()` for frequency estimation directly on data frames. It includes basically three parameters (for benchmarking issues a fourth parameter is provided) determining the data set, the key variables and the vector of sampling weights.

```
args(freqCalc)

   ## function (x, keyVars, w = NULL, alpha = 1)
   ## NULL
```

(for details, see `?freqCalc`).

In the following, these frequency counts are calculated with this function and the counts are added to the data set `eusilc`.

```
## information on frequencies are assigned to each observation
counts <- freqCalc(eusilc,
        keyVars = c("db040", "hsize", "pb220a"))$fk
## add counts to data
eusilc <- cbind(eusilc, counts)
## first 6 rows of the sample
head(eusilc[,c("db040","hsize", "pb220a", "counts")])

   ##   db040 hsize pb220a counts
   ## 1 Tyrol     3     AT    222
   ## 2 Tyrol     3  Other     47
   ## 3 Tyrol     3   <NA>    237
   ## 4 Tyrol     4     AT    387
   ## 5 Tyrol     4     AT    387
   ## 6 Tyrol     4   <NA>    408
```

The frequencies are assigned to each individual. But we simple can aggregate this information, i.e. to receive sample frequencies (aggregated information on cells) presented for each key; we can do this with the following code.

```
X <- aggregate(counts ~ db040 + hsize + pb220a, eusilc, mean)
## number of keys
nrow(X)

   ## [1] 164

## unique keys (frequency = 1)
sum(X$counts == 1)

   ## [1] 2

## first 6 counts for keys (out of 164)
head(X[, c("db040", "hsize", "pb220a", "counts")])

   ##           db040 hsize pb220a counts
   ## 1    Burgenland     1     AT     57
   ## 2     Carinthia     1     AT    114
   ## 3 Lower Austria     1     AT    319
   ## 4      Salzburg     1     AT     93
   ## 5        Styria     1     AT    254
   ## 6         Tyrol     1     AT    100
```

There exist 164 possible keys (considering the specified three categorical key variables) and two unique combinations. Function `head()` shows the first 10 keys of the `eusilc` data set with its corresponding frequency counts.

In Fig. 3.1 the cell size indices are visualized from the `eusilc` data set related to three categorical key variables. There are many cells with small frequency counts and only few that include more than 200 observations (see Fig. 3.1). The figure can be produced with the following code.

```
library(ggplot2)
hist <- qplot(X$counts, xlab="Cell size", ylab="Count") +
  geom_bar(fill="grey50")
print(hist)
```

So far, we done all sample frequency calculations using the first method (*current method*). However, in **sdcMicro** a general approach—the method that we called *category size*—is available for use when using **sdcMicro** version >4.6.1 onwards. The function `freqCalc` has a function parameter `alpha`, which is a numeric value between 0 and 1 specifying how much keys that contain missing values (NAs) should contribute to the calculation of sample and population frequency counts. For the default value of 1 (*current method*), nothing changes with respect to the implementation in prior versions of **sdcMicro**. Each wildcard-match would be counted, while for `alpha=0` keys with missing values would be basically ignored.

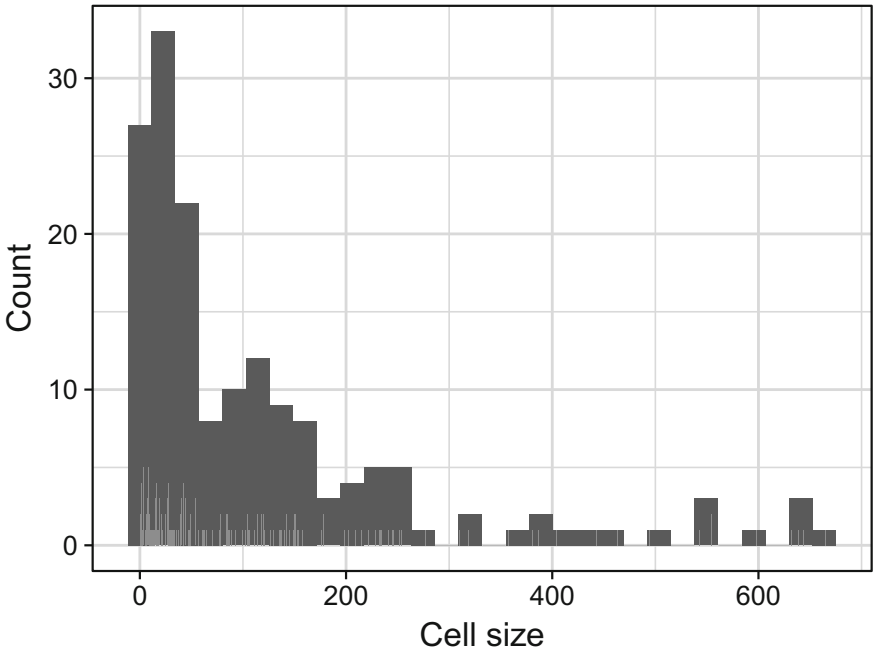


Fig. 3.1 Cell size indices of the Austrian EU-SILC data when selecting region, household size and citizenship as key variables

```
df <- data.frame("key1" = c(1,1,2,NA),
                 "key2" = c(1,1,1,1),
                 "key3" = c(3,NA,3,NA),
                 w = c(10,20,30,40))
f1 <- freqCalc(df, keyVars = 1:3, w = 4, alpha = 1)
f0 <- freqCalc(df, keyVars = 1:3, w = 4, alpha = 0)
f01 <- freqCalc(df, keyVars = 1:3, w= 4, alpha = 0.1)
d <- data.frame("f1"=f1$fk, "f0"=f0$fk, "f01"=f01$fk)
cbind(df, d, data.frame("F1"=f1$Fk, "F0"=f0$Fk, "F01"=f01$Fk))

   ##   key1 key2 key3  w f1 f0 f01  F1 F0 F01
   ## 1    1    1    3 10  3  1 1.2  70 10  16
   ## 2    1    1   NA 20  3  2 2.1  70 30  34
   ## 3    2    1    3 30  2  1 1.1  70 30  34
   ## 4   NA    1   NA 40  4  3 3.1 100 80  82
```

3.3 Principles of k-anonymity and l-diversity

Assuming that sample uniques are more likely to be re-identified, one way to protect confidentiality is to ensure that each distinct pattern of key variables is possessed by at least k records in the sample. This approach is called achieving k-anonymity (Samarati and Sweeney 1998; Samarati 2001; Sweeney 2002). More precisely, let $Z_1, ..., Z_q$ the categorical key variables of a data set with n records. Then k-anonymity is achieved if each possible combination of key variables contains at least k units in the microdata set, i.e. $f_j \geq k$ and $\forall j \in \{1, ..., n\}$.

A typical practice is to set $k = 3$, which ensures that the same pattern of key variables is possessed by at least three records in the sample. Using the previous notation, 3-anonymity means for all records.

Even if a group of observations fulfill k-anonymity, an intruder can be still discover sensitive information. For example, Table 3.1 satisfies 3-anonymity, given the two key variables gender and age. Suppose an intruder gets access to the sample inpatient records, however, and knows that his neighbor, a female in her twenties, recently went to the hospital. Since all records of females in their twenties have the same medical condition, the intruder discovers with certainty that his neighbor has cancer. In a different scenario, if the intruder has a male friend in his thirties who belongs to one of the first three records, the intruder knows that the incidence of his friend having heart disease is low and thus concludes that his friend has cancer.

The concept of l-diversity (Machanavajjhala et al. 2007) is used to address this limitation of k-anonymity. It was introduced as a stronger notion of privacy: a group of observations with the same pattern of key variables is l-diverse if it contains at least l "well-represented" values for the sensitive variable. Machanavajjhala et al. (2007)

Table 3.1 Example inpatient records illustrating k-anonymity and l-diversity

	Key variables		f_k	Sensitive variable	Distinct l-diversity
	Gender	Age group		Medical condition	
1	Male	30s	3	Cancer	2
2	Male	30s	3	Heart disease	2
3	Male	30s	3	Heart disease	2
4	Female	20s	3	Cancer	1
5	Female	20s	3	Cancer	1
6	Female	20s	3	Cancer	1

interpreted "well-represented" in a number of ways, and the simplest interpretation, distinct l-diversity, ensures that the sensitive variable has at least l distinct values for each group of observations with the same pattern of key variables. As shown in Table 3.1, the first three records are 2-diverse because they have two distinct values for the sensitive variable, medical condition.

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 sensitive variable is specified (using `createSdcObj`). Note that the measure can calculated at any time using `ldiversity(sdc)` with optional function parameters to select another sensitive 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:

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

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

Machanavajjhala et al. (2007) define three different l-diversity measures:

distinct l-diversity: as the simplest definition that ensures that at least l distinct values for the sensitive field in each key;

entropy l-diversity: as the most complex definition, which defines entropy of a key where the fraction of observations that have a sensitive value;

recursive l-diversity: as a compromise definition that ensures the most common sensitive value does not appear too often in a key while less common sensitive values are ensured not to appear too infrequently in the same key.

Additional, there is also a version implemented in **sdcMicro** when more than one sensitive variable is present in the data. This leads to multiple entropy and multi recursive l diversity measures. The result of all these measures are stored already in object `res1`, and the first six observations have the following l-diversity.

```
head(res1[,1:5])
```

```
##      income_Distinct_Ldiversity income_Entropy_Ldiversity
## [1,]                          7                   7.00000
## [2,]                          7                   7.00000
## [3,]                         19                  19.00000
## [4,]                         22                  21.65466
## [5,]                          5                   5.00000
## [6,]                          4                   4.00000
##      income_Recursive_Ldiversity MultiEntropy_Ldiversity
## [1,]                           7                       0
## [2,]                           7                       0
## [3,]                          19                       0
## [4,]                          21                       0
## [5,]                           5                       0
## [6,]                           4                       0
##      MultiRecursive_Ldiversity
## [1,]                         0
## [2,]                         0
## [3,]                         0
## [4,]                         0
## [5,]                         0
## [6,]                         0
```

For more information, see also the help files of **sdcMicro**.

Remark: k-anonymity depends on the chosen rules that determine how frequencies are estimated in case of missing values in the key variables. In Sect. 3.2.2 this has already been briefly touched, and five possible approaches how to estimate sample frequencies in case of missing values have been mentioned. This is further discussed in Sect. 4.2.2.

3.3.1 Simplified Estimation of Population Frequency Counts

Before more sophisticated methods are shown, the simplest approach to estimate population frequency counts is shown. For illustration, the toy data set given in Table 3.2 is used. In the following R code this data set is referred as `toyData`.

Of course, from Table 3.2 the direct identifiers have to be deleted before dissemination. In this demonstration only the variable `name` is a direct identifier. Let us fix `Gender` and `Occupation` as categorical key variables.

The population frequency counts are usually not known, since only few information is available about a population and typically not all key variables are present in the population.

Table 3.2 Toy data set (`toyData`) to illustrate the estimation of population frequency counts

	Name	Year of birth	Gender	Citizenship	Occupation	Income	Weight
1	Max Mustermann	1978	m	AUT	Worker	35,000	110
2	Josef Meier	1945	m	AUT	Pensioner	23,500	70
3	Sabine Schnuller	1991	w	AUT	Student	7000	80
4	John Doe	1966	m	US	Employee	41,200	120
5	Susan Rose	1989	w	AUT	Student	0	130
6	Markus Roller	1972	m	AUT	Employee	31,100	90
7	Christoph Valon	1944	m	AUT	Pensioner	21,400	150
8	Ulrike Mayer	1932	w	D	Pensioner	17,600	150
9	Stefan Fuchs	1992	m	AUT	Worker	27,500	130
10	Rainer Thomas	1950	m	AUT	Pensioner	25,700	150
11	Julia Gross	1976	w	AUT	Employee	37,000	140
12	Nadine Glatz	1987	w	AUT	Student	0	120
13	Makro Dilic	1990	m	AUT	Worker	21,050	90
14	Sandra Stadler	1941	w	AUT	Pensioner	28,500	80

It is often useful to assign the frequency counts to each observation. This information is automatically stored whenever an object of class *sdcMicroObj* is created. Note that also function `freqCalc` can be used to assign sample frequency counts to every observation of the data set.

Let us create first an object of class *sdcMicroObj*. Here we specify the key variables and provide the variable name for the vector of sampling weights.

```
sdcToy <- createSdcObj(toyData,
            keyVars = c("Gender", "Citizenship",
                        "Occupation"),
            numVars = "Income",
            weightVar = "Weight",
            )
```

As discussed by Willenborg and De Waal (2000) the simplest approach to estimate F_j under the assumption of simple random sampling without replacement is given by $\hat{F}_j = \frac{f_j}{f}$, where $f = \frac{n}{N}$ is the sampling fraction. In general for this approach, the sample frequency counts are multiplied by their sampling weights and summed up. The sample and population frequency counts are already estimated and included in `sdcToy`. Let us extract this information to further investigate how the population frequency counts are estimated for this data set.

```
toy2 <- cbind(toyData[,c(3:5,7)],
              get.sdcMicroObj(sdcToy,
                              "risk")$individual[,2:3])
toy2
```

```
##    Gender Citizenship Occupation Weight fk  Fk
## 1       m         AUT     Worker    110  3 330
## 2       m         AUT  Pensioner     70  3 370
## 3       w         AUT    Student     80  3 330
## 4       m          US   Employee    120  1 120
## 5       w         AUT    Student    130  3 330
## 6       m         AUT   Employee     90  1  90
## 7       m         AUT  Pensioner    150  3 370
## 8       w           D  Pensioner    150  1 150
## 9       m         AUT     Worker    130  3 330
## 10      m         AUT  Pensioner    150  3 370
## 11      w         AUT   Employee    140  1 140
## 12      w         AUT    Student    120  3 330
## 13      m         AUT     Worker     90  3 330
## 14      w         AUT  Pensioner     80  1  80
```

We see that observation 6 is unique in the sample and the estimated population frequency $\hat{F}_k = 90$, which equals the sampling weight of observation 6. In addition we can observe that for the key $Gender = m \times Citizenship = AUT \times Occupation = Pensioner$ the sample frequency is 3, but the population frequency is 370. The estimated population frequencies are obtained by summing up the sample weights for observations corresponding to the same key. Population frequencies for the previous mentioned key can therefore be estimated by summation over the corresponding sampling weights, w_2, w_7 and w_{10}. In summary, three observations with the key $Genderm \times CitizenshipAUT \times OccupationPensioner$ exist in the sample and 370 observations with this pattern (key) can be expected to exist in the population.

In practice this simplified estimator of population frequency counts will not provide workable solutions to be used for estimating disclosure risk for complex survey data, see discussion Willenborg and De Waal (2000). For example, when n is small and N is much higher then $f_j = 0$ implies $\hat{F}_j = 0$ and $f_j = 1$ implies $\hat{F}_j = w$, where w is the weight of every drawn observation. If the weights are known for every observation in the sample data set, then the population frequencies $\hat{F}_j$ are the sum of the weights of each record that has the same key combination. Thus $\hat{F}_j = \sum_{i \in |\mathbf{S}_j|} w_i$, where $|\mathbf{S}_j|$ is a subsample of $\mathbf{S}$ regarding key j and w_i are the weights of record $\mathbf{i}$ in subsample $\mathbf{S}_j$.

To give a toy example to explain this issue, we assume that **U** in the following code is our population.

```
set.seed(12)
U <- data.frame("var1" = sample(1:2, 16, replace=TRUE),
                "var2" = sample(1:3, 16, replace=TRUE))
```

From **U** we draw a sample with simple random sampling. A seed for the random number generator is set to stay reproducible. Since we used simple random sampling and we have drawn 4 out of 16 observations ($\pi_i = \frac{1}{4} \;\; i = 1, \ldots, 4$), we know the sampling weights ($w_i = \frac{1}{\pi_i}$) and assign it to the sample **S**.

```
set.seed(124)
select <- sample(1:nrow(U), 4)
S <- U[select, ]
S$weights = rep(4, nrow(S))
S

   ##   var1 var2 weights
   ## 2    2    2       4
   ## 7    1    1       4
   ## 8    2    3       4
   ## 6    1    3       4
```

Now the frequencies can be calculated for the sample (f_i) and estimated for the population (F_i).

```
f <- freqCalc(S, c("var1", "var2"), w=3)
cbind(S, fk=f$fk, Fk=f$Fk)

   ##   var1 var2 weights fk Fk
   ## 2    2    2       4  1  4
   ## 7    1    1       4  1  4
   ## 8    2    3       4  1  4
   ## 6    1    3       4  1  4
```

We see from this output that the estimated population frequency counts for the key ($var1 = 2 \;\wedge\; var2 = 3$) is 4 as for any other estimated population frequencies. Next we see the true frequency counts in the population (`Fktrue`). For the third line, the true frequency in the population is 1 and we highly overestimate this frequency.

```
Fktrue <- freqCalc(U, c("var1", "var2"))
cbind(U[select, ], Fk=f$Fk, Fktrue=Fktrue$Fk[select])

   ##   var1 var2 Fk Fktrue
   ## 2    2    2  4      2
   ## 7    1    1  4      6
   ## 8    2    3  4      1
   ## 6    1    3  4      3
```

In other words, for complex samples, especially for socio-economic samples with comparable values of weights, we would always overestimate small population

frequencies and if this estimation of population frequency will be the basis for risk estimation, the risk would highly be underestimated.

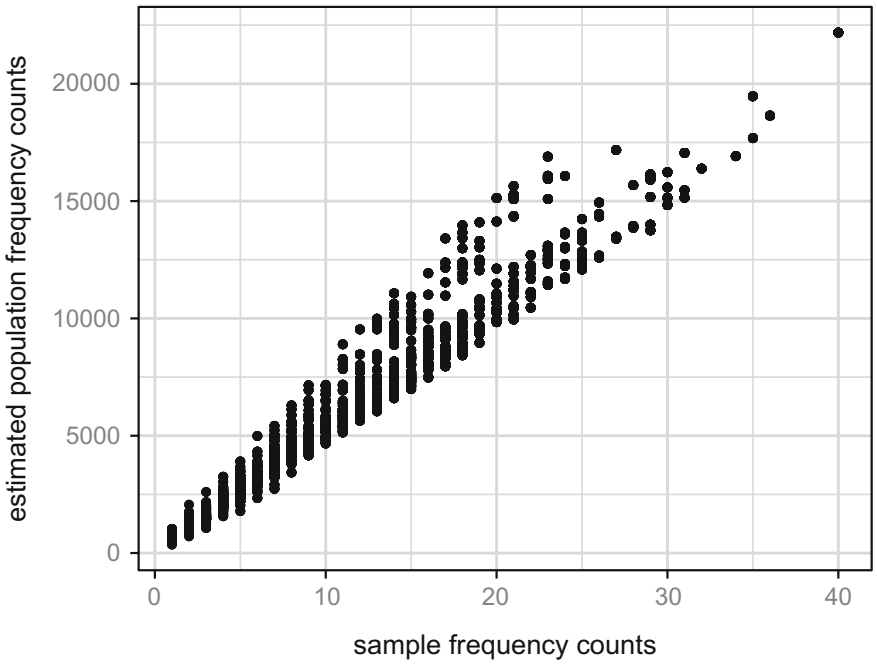
Of course, there is a relation between the sample frequency counts and the estimated population frequency counts, since the higher the sample frequencies, the more individuals contribute to the population frequencies. However, also the sampling weights play a role.

This relationship could also be shown using the following application. From the `eusilc` data set from package **laeken** the variables `age`, `pb220a` (citizenship) and `rb090` (gender) are selected as categorical key variables. First, we create an object of class *sdcMicroObj*.

```
library("laeken")
data(eusilc)
sdc <- createSdcObj(eusilc,
                    keyVars = c("age", "pb220a", "rb090", "db040"),
                    weightVar = "rb050")
```

We access the frequency counts and plot them.

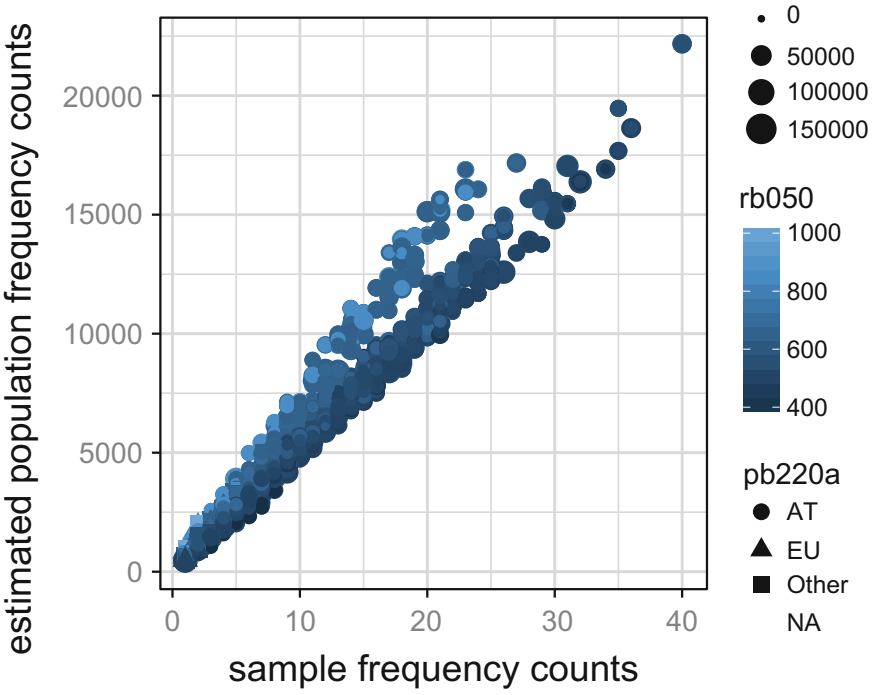
```
risk <- slot(sdc, "risk")$individual
freq <- data.frame(risk[, c("fk", "Fk")])

library("ggplot2")
gg <- ggplot(freq, aes(x=fk, y=Fk)) + geom_point() +
  xlab("sample frequency counts") +
  ylab("estimated population frequency counts")
print(gg)
```



We see that there is a linear trend, the higher the sample frequencies, the higher the population frequencies.

We can further investigate if some groups have lower frequencies, and we can observe the relation with income and the sampling weights.

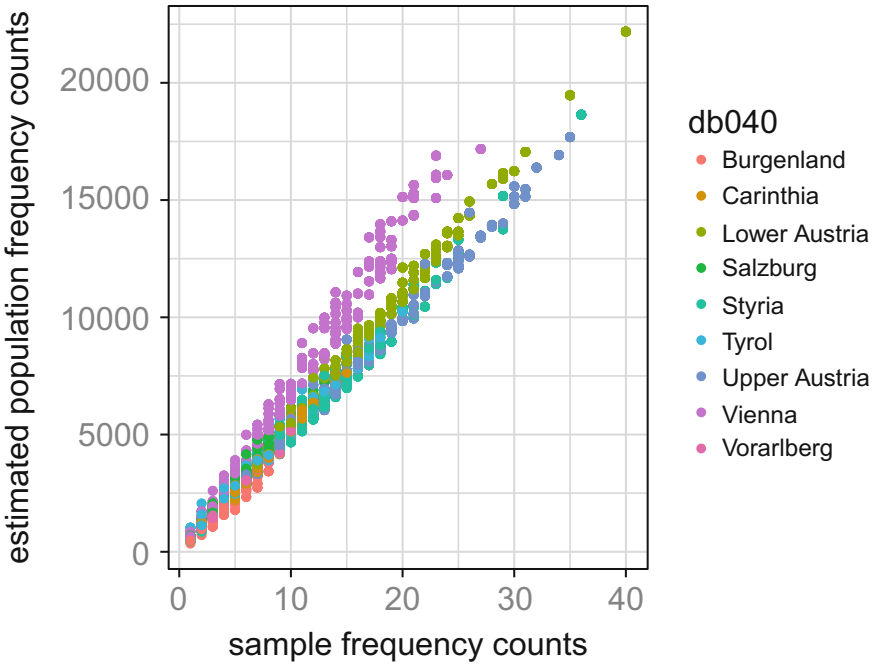
```
eusilc$fk <- freq$fk
eusilc$Fk <- freq$Fk
library(ggplot2)
gg <- ggplot(eusilc,
             aes(x=fk, y=Fk,
                 shape=pb220a, colour=rb050,
                 size=eqIncome)) +
             geom_point() +
             xlab("sample frequency counts") +
             ylab("estimated population frequency counts")
print(gg)
```



We see that individuals with higher incomes have generally lower frequencies according to the selected key variables. Moreover, the lower frequencies are more related to categories *EU* and *Others*. This is because *AT* is much more frequent than

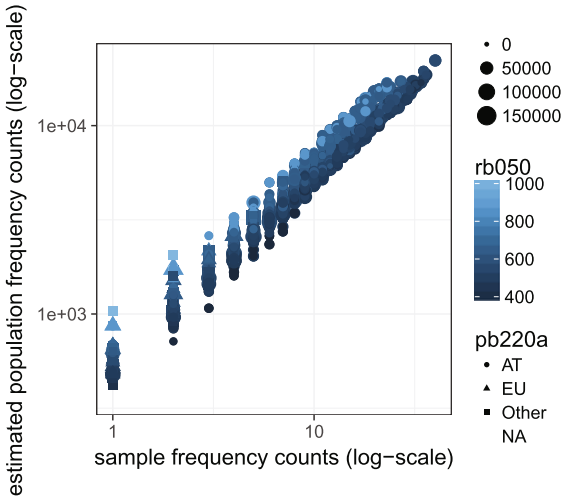
EU or *Others*. In addition, the sample design is indirectly reflected in this graphic. We can see this more clearly if we color the points depending on the region (`db040`) in the following graphic.

```
gg <- ggplot(eusilc, aes(x=fk, y=Fk, colour=db040)) +
            geom_point() + xlab("sample frequency counts") +
            ylab("estimated population frequency counts")
print(gg)
```



Lastly, we can look at log-transformed frequency counts. It is clearly visible that the estimated population frequencies depending on the high of the sampling weights.

```
gg <- ggplot(eusilc,
             aes(x=fk, y=Fk, shape=pb220a,
                 colour=rb050, size=eqIncome)) +
             geom_point() + scale_x_log10() + scale_y_log10() +
             xlab("sample frequency counts (log-scale)") +
             ylab("estimated population frequency counts (log-scale)")
print(gg)
```



Exercises:

Question 3.1 **Sample frequency counts versus estimated population frequency counts**
We now know how to calculate the sample frequency counts and already learned one possibility to estimate population frequency counts. Use the `eusilc` data set from package **laeken**. Assume the following disclosure scenario that defines `age`, `pb220a` (citizenship), `pl030` (education level), `rb090` (gender) and `hsize` (household size) as categorical key variables. Use the package **sdcMicro** to create an object of class *sdcMicroObj*. Access the sample and population frequency counts and plot them against each other. What can you observe?

3.4 Special Uniques Detection Algorithm (SUDA)

An alternative approach for defining disclosure risks is based on the concept of special uniqueness. For example, the eighth record in Table 3.3 is a sample unique with respect to the key variable set (i.e., {*age group*, *gender*, *income*, *education*}). Furthermore, a subset of the key variable set, for example, the combination of variables {*gender*, *education*} with the categories *male* and *university*}, is also unique in the sample. An observation is defined as a special unique with respect to a variable set Q, if it is sample unique both on Q and on a subset of Q (Elliott et al. 1998). Research has shown that special uniques are more likely to be population uniques than random uniques (Elliott et al. 2002).

Table 3.3 Example data set illustrating SUDA scores

	Age group	Gender	Income (k)	Education	f_k	SUDA score	Risk DIS-SUDA method
1	20s	Male	$\geq$50	High school	2	0	0
2	20s	Male	$\geq$50	High school	2	0	0
3	20s	Male	$\leq$50	High school	2	0	0
4	20s	Male	$\leq$50	High school	2	0	0
5	20s	Female	$\leq$50	University	1	1	0.0105
6	20s	Female	$\leq$50	High school	1	0.5	0.0046
7	20s	Female	$\leq$50	Middle school	1	1.75	0.0203
8	60s	Male	$\leq$50	University	1	2.25	0.0272

3.4.1 Minimal Sample Uniqueness

A set of computer algorithms, called SUDA, was designed to comprehensively detect and grade special uniques (Elliott et al. 2002). SUDA takes a two-step approach. In the first step, all unique attribute sets up to a user-specified size are located for each observation. SUDA considers only *Minimal Sample Uniques* (MSUs), which are unique variable sets without any unique subsets within a sample. In the example presented in Table 3.3, *male—university* is a MSU of observation 8 because none of its subsets, *male* or *university*, is unique in the sample. Also *60s* is a MSU in observation 8. An example for sample uniqueness but not being an MSU is the combination of *60s*, *male*, $\leq 50\,k$, *university*. This is a unique variable set, but not a MSU because its subsets (*60s*, *male*, *university*) and (*male*, *university*) are both unique subsets in the sample.

3.4.2 SUDA Scores

Once all MSUs have been found, a SUDA score is assigned to each observation indicating the risk using the size and distribution of MSUs within each observation (Elliott et al. 2002). The potential risk of the records is determined based on two issues:

1. the smaller the number of variables spanning the MSU within an observation, the higher the risk of the observation, and
2. the larger the number of MSUs in an observation, the higher the risk of the observation.

For each MSU of size k contained in a given observation, a score is computed by

$$s_i = \begin{cases} \frac{1}{q!}\prod_{i=k}^{M}(q-i), & \text{if } \; i \leq M. \\ 0, & \text{otherwise} \quad , \end{cases} \tag{3.3}$$

where M ($M > 0$) is the user-specified maximum size of MSUs, and q is the total number of categorical key variables in the data set. Note that if i is larger than M, the SUDA score should be set to zero. By definition, the smaller the size k of the MSU, the larger the score for the MSU (look at (q − i) in Eq. 3.3). The final SUDA score for the observation is computed by multiplying the scores for each MSU (have a look at the product in Eq. 3.3). In this way, observations with more MSUs are assigned a higher SUDA score.

The final SUDA score is calculated by normalizing these SUDA scores by dividing them by $q!$, with q being the number of key variables.

To illustrate how SUDA scores are calculated, we take a look at observation 8 in Table 3.3. This observation has two MSUs: *60s* of size 1, and (*male*, *university*) of size 2. Suppose the maximum size of MSUs is set at 3, the non-normalized score assigned to *60s* is computed by $\prod_{i=1}^{3}(4-i) = 6$, and the non-normalized score assigned to (*male*, *university*) is $\prod_{i=2}^{3}(4-i) = 2$. The normalized SUDA score is then obtained by normalizing the scores and by summation over these two normalized scores.

3.4.3 SUDA DIS Scores

In order to estimate observation-level disclosure risks, SUDA scores can be used in combination with the *Data Intrusion Simulation* (DIS) metric (Elliot and Manning 2003), which is a method for assessing a global disclosure risk for the entire data set. To receive the 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; we refer to Elliott et al. (2002), Elliot and Manning (2003) for details.

3.4.4 SUDA in sdcMicro

Both SUDA and DIS-SUDA scores can be computed using sdcMicro (Templ et al. 2015). Given that the implementation of SUDA can be computational demanding, sdcMicro uses an improved SUDA2 algorithm, which more effectively locates the boundaries of the search space for MSUs in the first step (Manning et al. 2008). Table 3.3 presents the observation-level risks estimated using the DIS-SUDA approach for the sample data set. Instead of replacing the risk measures introduced

in Sect. 3.5.1, the SUDA scores and DIS-SUDA approach can be best used as an enhancement of the k-anonymity approach (see Sect. 3.3). For example, compared to the risk measures presented in Table 3.3, for records with the same sample frequency count, the DIS-SUDA score (see also Table 3.3) does not fully account for the sampling weights, while the risk measures based on negative binomial model are typically lower for records with larger sampling weights.

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 built, 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. First we apply SUDA2 on a toy data set from Table 3.3.

```
tab <- data.frame("age" = c(rep("20s", 7), "60s"),
                  "gender" = c(rep("male", 4), rep("female", 3), "male"),
                  "income" = c("50k+", "50k+", rep("50k-", 6)),
                  "education" = c(rep("highschool", 4), "university",
                          "highschool", "middleschool", "university"))
su <- suda2(tab)
## print dis suda scores summary
su
```

```
##
## Dis suda scores table:
## - - - - - - - - - - -
##     Interval Number of records
## 1       == 0                 4
## 2 (0.0, 0.1]                 4
## 3 (0.1, 0.2]                 0
## 4 (0.2, 0.3]                 0
## 5 (0.3, 0.4]                 0
## 6 (0.4, 0.5]                 0
## 7 (0.5, 0.6]                 0
## 8 (0.6, 0.7]                 0
## 9      > 0.7                 0
## - - - - - - - - - - -
## Attribute contribution:
## - - - - - - - - - - -
##    variable contribution
## 1       age     40.90909
## 2    gender     27.27273
## 3    income      0.00000
## 4 education     68.18182
## - - - - - - - - - - -
```

We see that four observations has a considerable high risk. The individual SUDA scores and DIS scores are stored on the following list elements.

```
names(su)
```

```
## [1] "contributionPercent"
## [2] "score"
## [3] "disScore"
## [4] "attribute_contributions"
## [5] "attribute_level_contributions"
```

We have a look on the scores and dis SUDA scores.

```
su$score
```

```
## [1] 0.00 0.00 0.00 0.00 1.00 0.50 1.75 2.25
```

```
su$disScore
```

```
## [1] 0.000000000 0.000000000 0.000000000 0.000000000
## [5] 0.010460685 0.004580335 0.020271161 0.027212235
```

And we can evaluate which variables contributed most to these scores. For observation eight, this is, of course, variable age since it has a MSU of size 1 with respect to age that counts more than the MSU of size 2 regarding the combination of gender and university.

```
su$contributionPercent
```

```
##      age_contribution gender_contribution income_contribution
## [1,]        0.0000000           0.0000000                   0
## [2,]        0.0000000           0.0000000                   0
## [3,]        0.0000000           0.0000000                   0
## [4,]        0.0000000           0.0000000                   0
## [5,]        0.5000000           0.5000000                   0
## [6,]        0.0000000           1.0000000                   0
## [7,]        0.0000000           0.0000000                   0
## [8,]        0.7777778           0.2222222                   0
##      education_contribution
## [1,]              0.0000000
## [2,]              0.0000000
## [3,]              0.0000000
## [4,]              0.0000000
## [5,]              1.0000000
## [6,]              1.0000000
## [7,]              1.0000000
## [8,]              0.2222222
```

Let us resume by using a larger data set used before. Remember, we already created an *sdcMicroObj* called `sdc` from the data set `eusilc`. We apply SUDA directly on this object

```
sdc <- suda2(sdc)
```

The dis SUDA scores summary as well as all list elements can be extracted using `get.sdcMicroObj` or simply `slot`. We just look the results of the print method and obtain that 345 observations have higher dis SUDA score than 0.1.

```
su_silc <- slot(sdc, "risk")$suda
names(su_silc)
```

```
## [1] "contributionPercent"
## [2] "score"
## [3] "disScore"
## [4] "attribute_contributions"
## [5] "attribute_level_contributions"
```

```
su_silc
```

```
##
## Dis suda scores table:
## - - - - - - - - - - -
##     Interval Number of records
## 1       == 0             14482
## 2 (0.0, 0.1]               345
## 3 (0.1, 0.2]                 0
## 4 (0.2, 0.3]                 0
## 5 (0.3, 0.4]                 0
## 6 (0.4, 0.5]                 0
## 7 (0.5, 0.6]                 0
## 8 (0.6, 0.7]                 0
## 9      > 0.7                 0
## - - - - - - - - - - -
## Attribute contribution:
## - - - - - - - - - - -
##   variable contribution
## 1      age    100.00000
## 2   pb220a     64.37055
## 3    rb090     26.60333
## 4    db040     80.99762
## - - - - - - - - - - -
```

3.5 The Individual Risk Approach

To estimate the frequencies of the population F_k, it is assumed that the population is drawn from a superpopulation. In fact, this means that the frequencies in the population either will be generated synthetically by drawing from a specific distribution of the frequency counts or quantiles of the assumed distribution of F_k are used. Using quantiles of the prior assumed distribution of F_k makes it possible to estimate the risk of each statistical unit. However, this estimation is just as good as the frequency counts of the population are modeled and how well the model assumption are fulfilled. Many suggestions exist in literature: the use of a Poisson-Gamma superpopulation model (Bethlehem et al. 1990), the Dirichlet-multinomial model (Hoshino and Takemura 1998), the negative binomial model (Benedetti and Franconi 1998; Franconi and Polettini 2004), a log-linear model (Skinner and Holmes 1998; Skinner and Shlomo 2006), a multinominal model (Forster and Webb 2007), the Poisson-inverse Gaussian model (Carlson 2002a) and references therein.

The estimation procedure of sample counts given the population counts is in the following modeled by assuming a negative binomial distribution (see Rinott and Shlomo 2006). This is also implemented in **sdcMicro** (see Templ et al. 2015) and called by the **sdcMicroGUI** (Kowarik et al. 2013).

3.5.1 The Benedetti-Franconi Model for Risk Estimation

For the popular *Benedetti-Franconi Model* (Benedetti and Franconi 1998; Franconi and Polettini 2004) or sometimes colloquially referred as the *Italian approach*, $F_k|f_k$ has to be estimated, i.e. the frequency counts in the population given the frequency counts in the sample. A common assumption is $F_k \sim Poisson(N\pi_k)$ (independently) (see, e.g., Franconi and Polettini 2004), where N is assumed to be known and with π_k the inclusion probabilities. (Binomial) Sampling from F_k means that $f_k|F_k \sim Bin(F_k, \pi_k)$. By standard calculations (see, e.g., Bethlehem et al. 1990) one gets

$$f_k \sim Poisson(N\pi_k) \text{ and } F_k|f_k \sim f_k + Poisson(N(1-\pi_k)) \ .$$

Concerning the risk estimation, the uncertainty on the frequency counts of the population is accounted in a Bayesian fashion by assuming that the population frequency given the sample frequency, $F_k|f_k$, is drawn from a negative binomial distribution with success probabilities p_k and the number of successes f_k (Polletini and Seri 2004; Rinott and Shlomo 2006). By using this assumption Benedetti and Franconi (1998) estimated the risk τ_2 by the well known and so called "model from Benedetti and Franconi". Using this background, Capobianchi et al. (2001) estimated the individual risk $\hat{r}_k$ as follows

$$\hat{r}_k = \left(\frac{\hat{p}_k}{1-\hat{p}_k}\right)^{f_k} \left\{ A_0 \left(1 + \sum_{j=0}^{f_k-3} (-1)^{j+1} \prod_{l=0}^{j} B_l\right) + (-1)^{f_k} \log(\hat{p}_k) \right\} \ , \quad (3.4)$$

whereas

$$\hat{p}_k = \frac{f_k}{\hat{F}_k} = \frac{f_k}{\sum_{i \in \{j|x_{j.}=x_{k.}\}} \pi_i} \ ,$$

while

$$B_l = \frac{(f_k-1-l)^2}{(l+1)(f_k-2-l)} \frac{\hat{p}_k^{l+2-f_k}-1}{\hat{p}_k^{l+1-f_k}-1} \text{ and } A_0 = \frac{\hat{p}_k^{1-f_k}-1}{f_k-1} \ .$$

If $f_k = 1$ (Capobianchi et al. 2001) use

$$\hat{r}_k = \frac{\hat{p}_k}{1-\hat{p}_k} \log\left(\frac{1}{\hat{p}_k}\right) \ ,$$

while if $f_k = 2$ they use

$$\hat{r}_k = \frac{\hat{p}_k}{1-\hat{p}_k} - \left(\frac{\hat{p}_k}{1-\hat{p}_k}\right)^2 \log\left(\frac{1}{\hat{p}_k}\right) \quad .$$

If the sample is large the computation in Formula 3.4 becomes infeasible, but the following approximation gives workable approximations (Capobianchi et al. 2001):

$$\hat{r}_k = \frac{\hat{p}_k}{f_k - (1-\hat{p}_k)}$$

Using the toy data set from Table 3.3, we can compare the individual risk to the risk from SUDA.

```
r <- measure_risk(tab,
  keyVars=c("age","gender","income","education"))$Res
```

The risk is high for all observations, but especially for observation five to eight. The risk is the same for these observation. Is it an indication that the individual risk approach is not as good compared to SUDA, since SUDA showed more variety in risk for each observation. Naturally observation eight should have higher risk which is not the case for the individual risk approach. Not at all. Or in other words, only for this particular toy data set that not includes sampling weights (nor have a hierarchical structure). We can learn also by looking follow-up examples that SUDA is a well-defined concept for measuring risk for census data or data without complex sampling designs.

In general, the individual risk methods considers sampling weights, i.e. data that are collected using a complex sampling design. Let us do the calculations on a larger data set. We use the `testdata` from package **sdcMicro**.

```
sdc <- createSdcObj(testdata,
          keyVars=c('urbrur','water','sex','age'),
          numVars=c('expend','income','savings'),
          pramVars=c("walls"),
          w='sampling_weight',
          hhId='ori_hid')
```

The frequency counts can again be accessed by the following call, but also the estimated frequencies $\hat{F}_k$ explained in Sect. 3.3.1 and the individual risk is accessible. The risk as well as $\hat{F}_k$ is the same for each key k.

```
risk <- get.sdcMicroObj(sdc, type="risk")$individual
head(risk)
```

```
##              risk fk   Fk   hier_risk
## [1,] 1.663894e-03  7  700 0.004330996
## [2,] 1.663894e-03  7  700 0.004330996
## [3,] 5.552471e-04 19 1900 0.004330996
## [4,] 4.543389e-04 23 2300 0.004330996
## [5,] 2.493766e-03  5  500 0.009682082
## [6,] 3.322259e-03  4  400 0.009682082
```

However, if a data set also contains a hierarchical structure such as persons in households or employees in enterprises, we have to extend this approach, shown in the next section.

3.6 Disclosure Risks for Hierarchical Data

Many micro-data sets have hierarchical, or multilevel, structures; for example, individuals who are situated in households. Once an individual is re-identified, the data intruder may learn information about the other household members, too. It is important, therefore, to take into account the hierarchical structure of the data set when measuring disclosure risks. It is commonly assumed that the disclosure risk for a household is higher than or equal to the risk that at least one member of the household is re-identified. In any case, for hierarchical data, information collected at the higher hierarchical level (e.g., household level) would be equal for all individuals in the group belonging to that higher hierarchical level (e.g., household), e.g. household income or any household related information. This hierarchical structure creates a further level of disclosure risk because if one individual in the household is re-identified, the household structure allows for re-identification of the other household members in the same household. In addition, there is always information available that gives an indication which persons belong to the same household. This information can be included partly in the vector of sampling weights but often also this information is explicitly available in data sets (household ID).

A household-level disclosure risk can be estimated by subtracting the probability that no person from the household is re-identified from one. For example, if we consider a single household with three members, with individual disclosure risks of 0.1, 0.05 and 0.01, respectively, the disclosure risk for the entire household will be calculated as $1 - (1 - 0.1) \cdot (1 - 0.05) \cdot (1 - 0.01) = 0.15355$.

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`. The household related individual risks can be seen in the following output. It is already available since in the previous code block, the function parameter `hhId` was set, and the household risk was automatically estimated when calling function `createSdcObj`.

```
head(cbind("household-ID"=testdata$ori_hid, risk))
```

```
##      household-ID         risk fk   Fk   hier_risk
## [1,]            1 1.663894e-03  7  700 0.004330996
## [2,]            1 1.663894e-03  7  700 0.004330996
## [3,]            1 5.552471e-04 19 1900 0.004330996
## [4,]            1 4.543389e-04 23 2300 0.004330996
## [5,]            2 2.493766e-03  5  500 0.009682082
## [6,]            2 3.322259e-03  4  400 0.009682082
```

Persons in the same households receive the same risk.

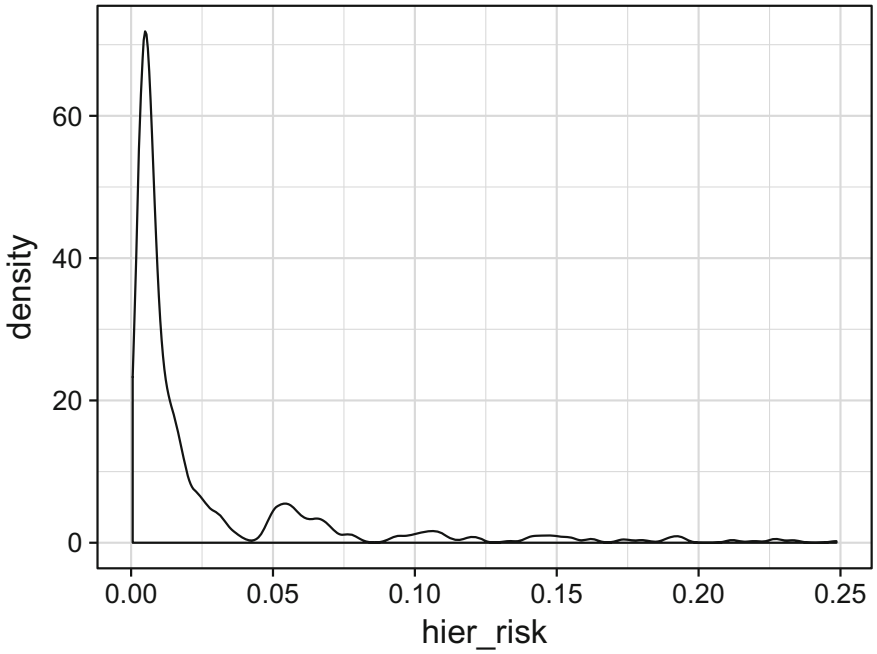


Fig. 3.2 Estimated density of the hierarchical risk

The distribution of the individual (hierarchical) risk is shown in Fig. 3.2.

It can be seen that quite some observations have high risk of disclosure, up to 0.25. Clearly, the risk should be reduced by applying SDC methods from Sect. 4.

Of course, there is also a relation between the risk and the estimated population frequencies, however this (negative) relation is not as strong (see Fig. 3.3), but in general this holds: the lower the sample and population frequency counts, the higher the individual risk.

Exercises:

Question 3.2 **Individual risk**
Take the `eusilc` data set from package **laeken**. Assume the following disclosure scenario that defines `age`, `pb220a` (citizenship), `pl030` (education level), `rb090` (gender) and `hsize` (household size) as categorical key variables. Use the package **sdcMicro** to create an object of class *sdcMicroObj* considering the sampling weights (function argument `weightVar` in `createSdcObj`) and the household ID (function argument `hhId`). Access the household risk and plot the distribution of the household risk. What can you detect? Are some estimated risks too high?

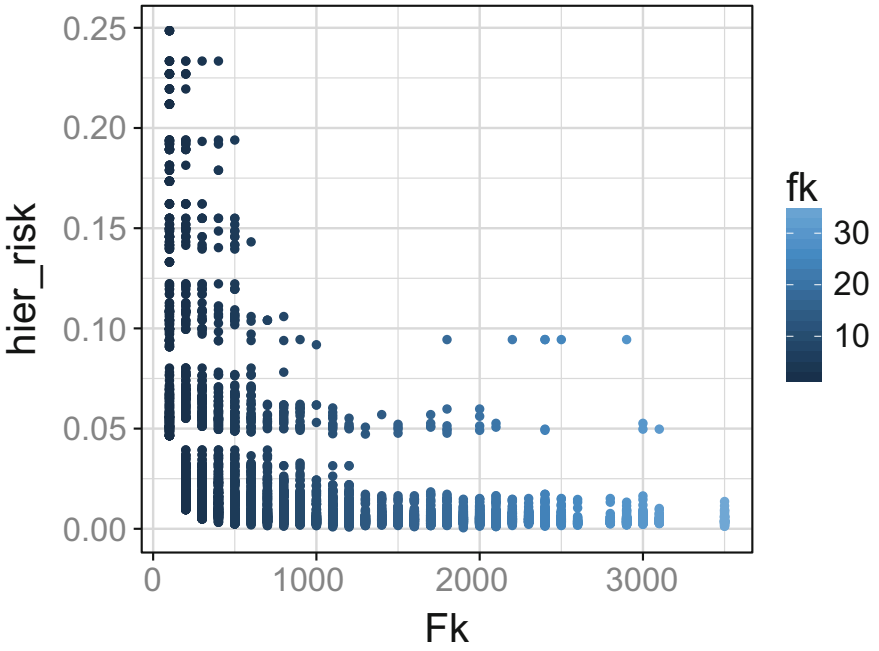


Fig. 3.3 Estimated population frequencies versus hierarchical risk

Question 3.3 **Individual risk versus household risk**
Take the same data and *sdcMicroObj* object created in the previous exercise. Access both the individual and the household risk and plot their distribution. What can you observe? Are the household risks higher than the individual risks?

3.7 Measuring Global Risks

In addition to record-level disclosure risk measures, a risk measure for the entire file-level or global risk micro-data set might be of interest. In this section, we present three common measures of global risks:

Expected number of re-identifications. The easiest measure of global risk is to sum up the record-level individual disclosure risks (defined in Sect. 3.5.1), which gives the expected number of re-identifications. Using the example from Sect. 3.6, the expected number of re-identifications is 117.2, the sum of the last column.

Global risk measure based on log-linear models. This measure, defined as the number of sample uniques that are also population uniques, is estimated using standard log-linear models (Skinner and Holmes 1998; Ichim 2008). The population frequency counts, or the number of units in the population that possess a specific pattern of key variables observed in the sample, are assumed to follow a Poisson distribution. The global risk can then be estimated by a standard log-linear model, using the main effects and interactions of key variables. A more precise definition is available in Skinner and Holmes (1998). The estimation of global risk using log-linear models is implemented in sdcMicro (Templ et al. 2015).

Benchmark approach. This measure counts the number of observations with record-level risks higher than a certain threshold and higher than the main part of the data. While the previous two measures indicate an overall re-identification risk for a microdata file, the benchmark approach is a relative measure that examines whether the distribution of record-level risks contains extreme values. For example, we can identify the number of records with individual risk satisfying the following conditions

$$r_i \geq 0.1 \wedge r_i \geq 2 \cdot (\tilde{\mathbf{r}} + 2 \cdot MAD(\mathbf{r})) \quad , \tag{3.5}$$

where $\mathbf{r}$ represents all record-level risks, and $MAD(\mathbf{r})$ is the median absolute deviation of all record-level risks.

Beneath is the print output of the corresponding function from **sdcMicro** showing both measures:

```
print(sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the
   data: 0
   ## Expected number of re-identifications: 24.78 (0.54
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications: 117.20 (2.56
   ##
   ------------------------------------------------------------------------
```

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

3.7.1 *Measuring the Global Risk Using Log-Linear Models*:

In this section model based methods to estimate population frequency counts are considered as described in Carlson (2002a, b), Skinner and Shlomo (2006). It is assumed that the cell frequencies are generated independently from Poisson distributions with individual rates λ_j, i.e. $F_j \sim Poisson(\lambda_j)$, $j \in \{1, ..., C\}$. This assumption holds if the sampling design is simple random sampling without replacement, then the distribution is hypergeometric with given size of the population N, number of categories C and inclusion probabilities π_j. If the number of cells is large enough each cell frequency may be approximated by a binomial distribution with parameters N and inclusion probability π_j. Since the population size is quite large and π_j small due to large C the Poisson distribution is used to approximate the binomial with $\lambda_j = N\pi_j$.

3.7.2 *Standard Log-Linear Model*

Log-linear models are used for modelling cell counts in contingency tables. These models declare how the expected cell count depends on levels of the categorical (key) variables. Let $\boldsymbol{\mu} = (\mu_1, ..., \mu_C)'$ denote the expected counts for the number of C cells of a contingency table. As in Agresti (2002) multidimensional log-linear models for positive Poisson means have the following form:

$$log(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\lambda} \quad , \tag{3.6}$$

where $log(\boldsymbol{\mu})$ is a $C \times 1$ vector containing the logarithms of the expected frequencies, $\mathbf{X}$ is a $C \times p$ model matrix and $\boldsymbol{\lambda}$ is a $p \times 1$ vector of model parameters.

3.7.3 *Clogg and Eliason Method*

As described in Clogg and Eliason (1987),Agresti (2002),Skinner and Shlomo (2006) the Clogg and Eliason approach additional considers the survey weights towards Eq. (3.6). They extend the log-linear model from Eq. 3.6 with an offset term $\mathbf{z} = (z_1, ..., z_C)'$ and $z_k = \frac{f_k}{\hat{F}_k}$, where $\hat{F}_k$ is the sum of survey weights across sample units in cell k. This consideration leads to the following adaption of the log-linear model:

$$log(\boldsymbol{\mu}) = log(\mathbf{z}) + \mathbf{X}\boldsymbol{\lambda} \quad . \tag{3.7}$$

3.7.4 Pseudo Maximum Likelihood Method

The fitted values for a linear model are solutions to the likelihood equations. We derive likelihood equations using Eq. (3.6) for a log-linear model. For a vector of frequency counts $\mathbf{f}$ with $\boldsymbol{\mu} = \mathbb{E}(\mathbf{f})$, the model is given by $log(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\lambda}$, for which $log(\mu_i) = \sum_j x_{ij} \cdot \lambda_j, \forall i \in \{1, ..., C\}$. The log likelihood for Poisson sampling is:

$$L(\boldsymbol{\mu}) = \sum_i f_i \cdot log(\mu_i) - \sum_i log(\mu_i) \quad . \tag{3.8}$$

Through Eqs. (3.6 and 3.8) the pseudo maximum likelihood approach yields the following equation:

$$log(\hat{\mathbf{F}}) = \mathbf{X}\boldsymbol{\lambda} \quad . \tag{3.9}$$

$\hat{F}_k$ is the sum of survey weights across sample units in cell k and $\hat{\mathbf{F}} = (\hat{F}_1, \hat{F}_2, ..., \hat{F}_C)'$.

3.7.5 Weighted Log-Linear Model

The weighted log-linear model is an extension of the standard log-linear model, that also considers the weights of each cell, i.e. the linear predictor for $\boldsymbol{\mu}$ also contains the weights as an explanatory variable. The weighted log-linear model is given by:

$$log(\boldsymbol{\mu}) = \tilde{\mathbf{X}}\boldsymbol{\lambda} \quad , \tag{3.10}$$

where $log(\boldsymbol{\mu})$ is a $C \times 1$ vector containing the logarithms of the expected frequencies, $\tilde{\mathbf{X}}$ is a $C \times q$ model matrix and $\boldsymbol{\lambda}$ is a $q \times 1$ vector of model parameters.

3.8 Application of the Log-Linear Models

To fit the log-linear models (standard, EC, PSE, weighted) the R function `glm()` of the standard package **stats** is used. The first and most important function argument of `glm()` is a formula specifying the response, predictors and possible interactions. In other words, from this formula, `glm()` builds a model (design) matrix and applies the (chosen family of) regression method on it. The following formulas are applied:

```
data(eusilc)
keyVars <- c("db040", "hsize", "rb090", "age", "pb220a", "pl030")
sdc <- createSdcObj(eusilc, keyVars = keyVars,
                    weightVar = "rb050", hhId = "db030")
form <- as.formula(paste(" ~ ", "db040 + hsize + rb090 +
             age + pb220a + age:rb090 + age:hsize +
             hsize:rb090"))
standardLLM <- as.formula(paste(c("fk",
                      as.character(form)),
                      collapse = ""))
standardLLM

   ## fk ~ db040 + hsize + rb090 + age + pb220a + age:rb090 + age:hsize +
   ##     hsize:rb090
   ## <environment: 0x117db9a60>

pseLLM <- as.formula(paste(c("Nk",
                      as.character(form)),
                      collapse = ""))
pseLLM

   ## Nk ~ db040 + hsize + rb090 + age + pb220a + age:rb090 + age:hsize +
   ##     hsize:rb090
   ## <environment: 0x117db9a60>

weightedLLM <- as.formula(paste(c("fk",
                    as.character(as.formula(
                      paste(c(form,"Fk"),
                            collapse="+")))),
                            collapse = ""))
weightedLLM

   ## fk ~ db040 + hsize + rb090 + age + pb220a + age:rb090 + age:hsize +
   ##     hsize:rb090 + Fk
   ## <environment: 0x117db9a60>
```

The vector named `keyVars` includes the considered categorical key variables. `standardLLM`, `pseLLM` and `weightedLLM` describe the model to be fitted. The predictor has the form `response ~ predictors`. For example, a specification of the form `age:rb090` indicates the interaction for all categories of the predictors `age` and `rb090`. This 2-way intercation model performs best for this disclosure risk scenario. For the EC approach the formula `standardLLM` is used and the offset term in function `glm()` is set to `offset` $= \frac{f_k}{\hat{F}_k}$. $\hat{F}_k$ are the estimated population frequency counts. They are calculated as the sum of weights across sample units in cell k. $\hat{F}_k$ is the response for the PSE model.

Next we need the frequency counts for each key. Remember that any function of **sdcMicro** (`freqCalc`, `createSdcObj`, `measure_risk`) assigns the frequencies to each observations. Therefore, the first action is to aggregate them to have it for each key.

```
## get frequencies
fk <- freqCalc(eusilc, keyVars, w="rb050")
## assign it to the data set
eusilc$fk <- as.numeric(fk$fk)
eusilc$Fk <- as.numeric(fk$Fk)
## aggregate, to have it for each key only
mu <- aggregate(fk ~ hsize + rb090 + age + db040 + pb220a + pl030,
                eusilc, unique)
## aggregate the weights
Fk <- aggregate(Fk ~ hsize + rb090 + age + db040 + pb220a + pl030,
                eusilc, unique)
## save it in a new data.frame
counts <- data.frame(mu, Fk=Fk$Fk)
#counts <- counts[,c(keyVars,"fk","weights")]
counts$age <- as.numeric(counts$age)
head(counts)

   ##   hsize rb090 age      db040 pb220a pl030 fk       Fk
   ## 1     5  male  16 Burgenland     AT     1  2 715.7143
   ## 2     7  male  16 Burgenland     AT     1  1 554.5000
   ## 3     4  male  18 Burgenland     AT     1  2 997.1515
   ## 4     4  male  19 Burgenland     AT     1  1 498.5758
   ## 5     4  male  20 Burgenland     AT     1  2 997.1515
   ## 6     5  male  20 Burgenland     AT     1  1 357.8571
```

The standard model can now be estimated as seen in the following code.

```
mod_standard <- glm(standardLLM, data = counts, family = poisson())
lambda_standard <- fitted(mod_standard)
summary(lambda_standard)

   ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   ##  0.5746  1.5340  1.8870  1.8920  2.2610  3.7250
```

The fitted parameters and test statistics of the coefficients are presented in Table 3.4. It is clear to see that the intercept is significantly non-zero. The p-value of the federal states is in some cases not significant, but in general the variable corresponding to federal state (`db040`) has a significant contribution. The variables `rb090` (gender) and `age` are not significant. The contribution of the variables `hsize` (household size) and `pb220a` (citizenship) is statistically significant at $\alpha = 0.05$.

The Clogg and Eliason method uses an offset term, which is defined as log-ratios of sample and estimated population frequency counts.

```
EC <- counts$fk/counts$Fk
EC <- log(EC + 0.1)
mod_EC <- glm(standardLLM, data = counts, family = poisson(), offset = EC)
lambda_EC <- fitted(mod_EC)
summary(lambda_EC)

   ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   ##  0.5734  1.5340  1.8870  1.8920  2.2620  3.7220
```

Table 3.4 Fitted regression coefficients, standard errors, value of the test statistics and p-value from the standard log-linear model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.4180	0.0829	5.04	0.0000
db040Carinthia	0.1020	0.0568	1.79	0.0728
db040Lower Austria	0.3812	0.0503	7.57	0.0000
db040Salzburg	0.0122	0.0585	0.21	0.8346
db040Styria	0.2380	0.0514	4.63	0.0000
db040Tyrol	0.1381	0.0555	2.49	0.0129
db040Upper Austria	0.3494	0.0506	6.91	0.0000
db040Vienna	0.3009	0.0515	5.84	0.0000
db040Vorarlberg	0.0527	0.0625	0.84	0.3986
hsize	0.0244	0.0167	1.46	0.1435
rb090female	−0.2620	0.0736	−3.56	0.0004
age	0.0070	0.0012	5.78	0.0000
pb220aEU	−0.6653	0.0605	−11.00	0.0000
pb220aOther	−0.5766	0.0383	−15.06	0.0000
rb090female:age	0.0021	0.0011	2.03	0.0424
hsize:age	−0.0018	0.0003	−5.49	0.0000
hsize:rb090female	−0.0114	0.0127	−0.90	0.3706

The summary of the regression model of the Clogg and Eliason method is shown in Table 3.5. The results are comparable to the standard method.

The pseudo Likelihood method uses the scaled population frequencies.

```
## Pseudo Likelihood: Nk ~ keyVars
f <- sum(counts$fk) / sum(counts$Fk)
N_k <- round(counts$Fk * f) #round
counts <- data.frame(counts, Nk = N_k)
mod_pse <- glm(pseLLM, data = counts, family = poisson())
lambda_pse <- fitted(mod_pse)
summary(lambda_pse)

   ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   ##  0.5235  1.5090  1.8950  1.9420  2.3390  4.3440
```

It's summary is shown in Table 3.6. The only difference to the standard and Clogg and Eliason method is in the significance in the coefficients belonging to the federal states (`db040`), which is now mostly highly significant.

Table 3.5 Fitted regression coefficients, standard errors, value of the test statistics and p-value from the Clogg and Eliason method

	Estimate	Std. error	z value	Pr(>|z|)
(Intercept)	2.7009	0.0829	32.58	0.0000
db040Carinthia	0.1042	0.0568	1.83	0.0669
db040Lower Austria	0.3843	0.0503	7.63	0.0000
db040Salzburg	0.0162	0.0585	0.28	0.7816
db040Styria	0.2397	0.0514	4.67	0.0000
db040Tyrol	0.1406	0.0555	2.53	0.0113
db040Upper Austria	0.3510	0.0506	6.94	0.0000
db040Vienna	0.3070	0.0515	5.96	0.0000
db040Vorarlberg	0.0547	0.0625	0.88	0.3813
hsize	0.0240	0.0167	1.44	0.1506
rb090female	−0.2621	0.0737	−3.56	0.0004
age	0.0071	0.0012	5.79	0.0000
pb220aEU	−0.6653	0.0605	−11.00	0.0000
pb220aOther	−0.5764	0.0383	−15.05	0.0000
rb090female:age	0.0021	0.0011	2.04	0.0418
hsize:age	−0.0018	0.0003	−5.51	0.0000
hsize:rb090female	−0.0114	0.0127	−0.90	0.3692

The weighted log-linear version (see the results in Table 3.7) do have the summed weights (population frequency counts) as one of the predictors. The results differs from the previous methods, and only the intercept, citizenship (`pb220a`) and the sum of weights (`Fk`) are significant.

```
mod_w <- glm(weightedLLM, data = counts, family = poisson())
lambda_w <- fitted(mod_w)
summary(lambda_w)

   ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   ##   0.868   1.344   1.456   1.892   1.831  44.880
```

The estimated $\hat{\lambda}_k,\ k = 1, ..., C$ are input to the global risk measures explained in the following section.

Table 3.6 Fitted regression coefficients, standard errors, value of the test statistics and p-value from the pseudo maximum likelihood method

	Estimate	Std. error	z value	Pr(>|z|)
(Intercept)	0.3952	0.0829	4.76	0.0000
db040Carinthia	0.1580	0.0579	2.73	0.0064
db040Lower Austria	0.4539	0.0515	8.82	0.0000
db040Salzburg	0.0867	0.0594	1.46	0.1443
db040Styria	0.2683	0.0527	5.09	0.0000
db040Tyrol	0.2115	0.0565	3.74	0.0002
db040Upper Austria	0.3807	0.0519	7.34	0.0000
db040Vienna	0.5517	0.0518	10.66	0.0000
db040Vorarlberg	0.1127	0.0635	1.78	0.0757
hsize	0.0127	0.0167	0.76	0.4469
rb090female	−0.2605	0.0728	−3.58	0.0003
age	0.0073	0.0012	6.05	0.0000
pb220aEU	−0.6686	0.0588	−11.36	0.0000
pb220aOther	−0.5886	0.0377	−15.60	0.0000
rb090female:age	0.0028	0.0010	2.69	0.0071
hsize:age	−0.0020	0.0003	−5.96	0.0000
hsize:rb090female	−0.0170	0.0127	−1.34	0.1815

3.9 Global Risk Measures

At this point we consider F_j as a stochastic variable without specific distribution assumptions. A measure of identification risk is given by

$$\mathbb{E}(1/F_j) = \sum_{i \in \mathbb{N}} \frac{1}{i} \mathbb{P}(F_j = i) \quad , \tag{3.11}$$

where $\mathbb{P}(F_j = i)$ denotes the probability that $F_j = i$, with $i = \{1, 2, ..., N\}$. If $i = 1$, we receive the probability of population uniqueness $\mathbb{P}(F_j = 1)$, which is the first term in the sum in (3.11).

As mentioned above we consider a random sample **S** of a finite population **U** of size N. The sample data is available to the intruder. Let f_j be the sample frequency counts. This leads to two measures of interest as described in Skinner and Shlomo (2006):

$$m_1 = \mathbb{E}(1/F_j | f_j) \quad , \tag{3.12}$$

Table 3.7 Fitted regression coefficients, standard errors, value of the test statistics and p-value from the weighted log-linear method

	Estimate	Std. error	z value	Pr(>|z|)
(Intercept)	0.1371	0.0846	1.62	0.1052
db040Carinthia	−0.0016	0.0569	−0.03	0.9782
db040Lower Austria	−0.0327	0.0511	−0.64	0.5229
db040Salzburg	−0.0950	0.0586	−1.62	0.1047
db040Styria	0.0440	0.0515	0.85	0.3929
db040Tyrol	−0.0147	0.0556	−0.26	0.7921
db040Upper Austria	0.0620	0.0508	1.22	0.2229
db040Vienna	−0.2285	0.0527	−4.33	0.0000
db040Vorarlberg	−0.0314	0.0625	−0.50	0.6157
hsize	0.0031	0.0171	0.18	0.8570
rb090female	0.0013	0.0749	0.02	0.9859
age	−0.0007	0.0013	−0.53	0.5948
pb220aEU	−0.2705	0.0609	−4.44	0.0000
pb220aOther	−0.2369	0.0389	−6.09	0.0000
Fk	0.0004	0.0000	72.29	0.0000
rb090female:age	−0.0002	0.0011	−0.16	0.8707
hsize:age	0.0002	0.0003	0.57	0.5662
hsize:rb090female	−0.0096	0.0128	−0.75	0.4533

$$m_2 = \mathbb{P}(F_j = 1 | f_j) \quad . \tag{3.13}$$

Under random sampling the pairs (F_j, f_j) are independent and the first measure (3.12) is the conditional expectation of $1/F_j$ and second (3.13) the conditional probability that $F_j = 1$ given f_j. When $f_j = 1$, (3.12) is highest, which is the worst case. Additionally the following holds for (3.13):

$$\mathbb{P}(F_j = 1 | f_j = i) = \begin{cases} \in [0, 1], & \text{if } i = 1 \\ 0, & \text{if } i \geq 2 \end{cases}$$

Considering the worst case, i.e. $f_j = 1$, leads to the focus on the following measures:

$$m_{1j} = \mathbb{P}(F_j = 1 | f_j = 1) \quad , \tag{3.14}$$

$$m_{2j} = \mathbb{E}(1/F_j | f_j = 1) \quad . \tag{3.15}$$

The measures given in Eqs. (3.14 and 3.15) are per observation measures and their values can vary between observations. Observation-level measures are discussed above. In the following, a measure for the global risk is described. This leads to consideration of aggregating observation-level measures given by

$$\hat{\tau}_1 = \sum_{\{j:f_j=1\}} m_{1j} = \sum_{\{j:f_j=1\}} \mathbb{P}(F_j = 1 | f_j = 1) \quad , \tag{3.16}$$

$$\hat{\tau}_2 = \sum_{\{j:f_j=1\}} m_{2j} = \sum_{\{j:f_j=1\}} \mathbb{E}(1/F_j | f_j = 1) \quad . \tag{3.17}$$

The global risk measure $\hat{\tau}_1$ is the expected number of sample uniques that are population unique and $\hat{\tau}_2$ is the expected number of correct matches for sample uniques (Skinner and Shlomo 2006). If the count of combinations C is large, $\hat{\tau}_1$ will closely approximate τ_1,

$$\hat{\tau}_1 \overset{C\to\infty}{\longrightarrow} \tau_1 = \sum_{j\geq 1} \mathbb{1}(f_j = 1, F_j = 1) \quad , \tag{3.18}$$

The same holds for $\hat{\tau}_2$ with:

$$\hat{\tau}_2 \overset{C\to\infty}{\longrightarrow} \tau_2 = \sum_{j\geq 1} \frac{\mathbb{1}(f_j = 1)}{F_j} \quad . \tag{3.19}$$

The population consists of N entities and the key divides the population into C cells. Each cell j is assigned a parameter $\rho_j > 0$ satisfying $\sum_{j=1}^{C} \rho_j = 1$ and a random independent variable F_j which is the population frequency in the cell j. With the assumption that $F_j \sim Poisson(\lambda_j)$, with $\lambda_j = N\rho_j$ and $j \in \{1, ..., C\}$, the following probability is given

$$\mathbb{P}(F_j = i) = \frac{\lambda_j^i e^{-\lambda_j}}{i!}, \; i \in \{0, 1, 2, 3, ...\} \quad . \tag{3.20}$$

The mean and variance of the random variables F_j is both equal to λ_j. It is also assumed that $f_j | F_j \sim Binomial(F_j, \pi_j)$, whereby π_j is the inclusion probability. Note that a sample drawn using Bernoulli sampling on a Poisson distributed population will remain Poisson.

For the sample frequency counts holds $f_j \sim Poisson(\lambda_j \pi_j)$. To estimate the number of sample uniques that are population unique the following probability has to be calculated

$$\mathbb{P}(F_j = 1 | f_j = 1) = e^{-\lambda_j(1-\pi_j)} \quad . \tag{3.21}$$

For the estimated risk measures $\hat{\tau}_1$ and $\hat{\tau}_2$ the following holds under the assumption of Poisson distribution

$$\hat{\tau}_1 = \sum_j \mathbb{1}(f_j = 1)\mathbb{P}(F_j = 1 | f_j = 1) = \sum_{\{j:f_j=1\}} e^{-\lambda_j(1-\pi_j)} \quad , \tag{3.22}$$

$$\hat{\tau}_2 = \sum_j \mathbb{E}(\frac{1}{F_j} | f_j = 1) = \sum_{\{j:f_j=1\}} \frac{1 - e^{-\lambda_j(1-\pi_j)}}{\lambda_j(1-\pi_j)} \quad . \tag{3.23}$$

With the following code the risk measures can be estimated. The function `modRisk` includes also all steps shown in the previous code chunks.

```
## risk for unmodified data using six key variables
m1 <- modRisk(sdc, method = "default", weights = eusilc$rb050,
         formulaM = form, bound = 5)@risk$model[1:2]
m2 <- modRisk(sdc, method = "CE", weights = eusilc$rb050,
         formulaM = form, bound = 5)@risk$model[1:2]
m3 <- modRisk(sdc, method = "PML", weights = eusilc$rb050,
         formulaM = form, bound = 5)@risk$model[1:2]
m4 <- modRisk(sdc, method = "weightedLLM", weights = eusilc$rb050,
         formulaM = form, bound = 5)@risk$model[1:2]
modelrisk <- data.frame(
               "method" = c("standard", "CE", "PML", "weightedLLM"),
               "tau1" = c(m1$gr1, m2$gr1, m3$gr1, m4$gr1),
               "tau2" = c(m1$gr2, m2$gr2, m3$gr2, m4$gr2))
modelrisk

##        method      tau1      tau2
## 1    standard 0.2964092 0.3604305
## 2          CE 0.2952641 0.3596631
## 3         PML 0.2960892 0.3596322
## 4 weightedLLM 0.3241130 0.3829982
```

Since we have 4109 uniques (27,78% of the data), the risk should be high. This is true for both measures $\hat{\tau}_1$ and $\hat{\tau}_1$. The risk is higher but comparable through all methods for $\hat{\tau}_2$.

Let us compare these risk estimates with previous risk approaches, the k-anonymity approach, the individual risk approach and SUDA2.

The percentage of observations violating 2-anonymity is 27.713% (4109 observations) and 3-anonymity is 46.854% (6947); see `print(sdc)` for details. This is also reflected in the SUDA scores table:

```
sdc <- suda2(sdc)
slot(sdc, "risk")$suda2
```

```
##
## Dis suda scores table:
## - - - - - - - - - - -
##     Interval Number of records
## 1       == 0             10745
## 2 (0.0, 0.1]              4058
## 3 (0.1, 0.2]                22
## 4 (0.2, 0.3]                 2
## 5 (0.3, 0.4]                 0
## 6 (0.4, 0.5]                 0
## 7 (0.5, 0.6]                 0
## 8 (0.6, 0.7]                 0
## 9      > 0.7                 0
## - - - - - - - - - - -
## Attribute contribution:
## - - - - - - - - - - -
##   variable contribution
## 1    db040     66.98131
## 2    hsize     64.66156
## 3    rb090     29.92720
## 4      age     95.66937
## 5   pb220a     31.54173
## 6    pl030     57.12919
## - - - - - - - - - - -
```

However, the individual risk is

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

```
## Risk measures:
##
## Number of observations with higher risk than the main part of the
data: 0
## Expected number of re-identifications: 57.49 (0.39
##
## Information on hierarchical risk:
## Expected number of re-identifications: 199.16 (1.34
##
----------------------------------------------------------------------
```

This is much smaller than the risk estimated by log-linear modelling. Because of the high amount of uniques, the individual risk may underestimate the true risk in this case.

3.10 Quality of the Risk Measures Under Different Sampling Designs

Surveys almost always are not drawn with simple random sampling.

The assumptions that $F_j \sim Poisson(\lambda_j)$ and that the λ_j fit the log-linear model are unaffected by a complex sampling scheme (Skinner and Shlomo 2006). However, if the sampling scheme is not SRS the risk measures $m_{1j} = e^{-\lambda_j(1-\pi_j)}$ and $m_{2j} = \frac{1-e^{-\lambda_j(1-\pi_j)}}{\lambda_j(1-\pi_j)}$ may be affected. But these expressions still hold if $\mathbb{P}(f_j = 1|F_j) = F_j\pi_j(1-\pi_j)^{F_j-1}$. In general an usable approximation $\mathbb{P}(f_j = 1|F_j) \approx F_j\pi_j(1-\pi_j)^{F_j-1}$ provides good results.

The models are tested by Totter (2015) with four different sampling designs (equal stratified sampling, proportional oversampling, proportional stratified sampling, simple random sampling) and three disclosure risk scenarios with different amounts of keys. From a close-to-reality population samples are drawn with this design. Knowing the true risk, the risk estimates from the samples have been evaluated.

The models sometimes underestimates the true disclosure risk depending on the disclosure scenario. One important point for good model performances is to choose a well-defined good interaction model (see also Skinner and Shlomo 2006). If the quality of the model is weak and/or there are too few predictors the disclosure risk will be underestimated. Model selection, i.e. to find a good fitting model with high predictive power, is therefore a crucial part of any risk estimation using log-linear models. Another criterion is the amount of keys, whereby a high amount of keys will generally give better results. All in all the standard method, the Eliason-Clogg and the pseudo maximum likelihood approach perform best and yield nearly the same results with simple random sampling, equal stratified sampling and proportional stratified sampling. The weighted log-linear model performs worst. Depending on the disclosure scenario, a few times the pseudo maximum method turned out to give worse predictions of disclosure risk.

Exercises:

Question 3.4 Estimate the global risk for data set `eusilc` as done above. Then, use a subset of `eusilc` and compare the results on the global disclosure risk. Do smaller data sets imply higher risks?

Question 3.5 Compare other risk measures such as the global risk estimated from the household risks. Does a smaller data set still imply a higher risk?

3.11 Disclosure Risk for Continuous Variables

The concept of uniqueness might not be applicable to continuous key variables, especially those with an infinite range, since almost every record in the data set will then be identified as unique. In this case, a more applicable method is to assess risk based on record linkages.

Assume a disclosure scenario where an intruder has access to a data set that has been perturbed before release, as well as an external data source that contains information on the same respondents included in the released data set. Because the original values of the released data set have been perturbed, the intruder attempts to match records in the released data set with those in the external data set using common variables. Assume that the external data source, to which the intruder has access, is the original data file of the released data set. Essentially, the record linkage approach assesses to what extent records in the perturbed data file can be correctly matched with those in the original data file. There are three general approaches to record linkage:

Distance-based record linkage. Pagliuca and Seri (1999) computes distances between records in the original data set and the protected data set. Suppose we have obtained a protected data set A' after applying some SDC methods to the original data set A. For each record in the protected data set A, we compute its distance to every record in the original data set, and consider the nearest and the second nearest records. Suppose we have identified x_1 and x_2 from the original data set as the nearest and second-nearest records, respectively, from record x_i in the protected data set. If x_1 is the original record used to generate x_i, or, in other words, record x_i in the protected data set A' and in the original data set A refer to the same respondent, then we mark record x_i "linked". Similarly, if record x_i was generated from r_2 (the second-nearest record in the original data set), we mark x_i "linked to the 2nd nearest". We proceed the same way for every record in the protected data set A'. Finally, disclosure risk is defined as the percentage of records in the protected data set A' marked as "linked" or "linked to the 2nd nearest". This record-linkage approach based on distance is compute-intensive and thus might not be applicable for large data sets.

Probabilistic record linkage. Alternatively, probabilistic record linkage (Jaro 1989) pairs records in the original and protected data sets, and uses an algorithm to assign a weight for each pair that indicates the likelihood that the two records refer to the same respondent. Pairs with weights higher than a specific threshold are labeled as "linked", and the percentage of records in the protected data marked as "linked" is the disclosure risk.

Interval disclosure. In addition, a third risk measure is called interval disclosure (Pagliuca and Seri 1999), which simplifies the distance-based record linkage and thus is more applicable for large data sets. In this approach, after applying SDC methods to the original values, we construct an interval around each masked

value. The width of the interval is based on the rank of the value the variable takes on or its standard deviation. We then examine whether the original value of the variable falls within the interval. The measure of disclosure risk is the proportion of original values that fall into the interval, assuming a worst case scenario that any original value that falls within the interval presents a correct match (or refers to the same respondent as) the masked value.

The latter approach is also described in Mateo-Sanz et al. (2004).The length of the intervals based on the standard deviation of the variable under consideration is calculated (see Fig. 3.4, upper left graphic; the boxes express the intervals).

Distance-based risks for continuous key variables are automatically estimated for objects of class `sdcMicroObj` using function `dRisk()`.

Let us create again an object of class *sdcMicroObj*. Note that continuous key variables have to be specified.

```
sdc <- createSdcObj(testdata,
          keyVars=c('urbrur','water','sex','age'),
          numVars=c('expend','income','savings'),
          pramVars=c("walls"),
          w='sampling_weight',
          hhId='ori_hid')
```

Current risks can be printed with:

```
print(sdc, "numrisk")

   ## Numerical key variables: expend, income, savings
   ##
   ## Disclosure risk is currently between [0.00
   ##
   ## Current Information Loss:
   ##   - IL1: 0.00
   ##   - Difference of Eigenvalues: 0.000
   ## ------------------------------------------------------------------------
```

This reports the percentage of observations falling within an interval centered on its masked value whereas 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 on the continuous key variables, the disclosure risk can be high (up to 100%) (and the information loss, discussed later in Chap. 5), is 0.

After applying a perturbation method (details on methods can be found in the next chapter), the risk is smaller.

```
sdc <- microaggregation(sdc)
print(sdc, "numrisk")

   ## Numerical key variables: expend, income, savings
   ##
   ## Disclosure risk is currently between [0.00
   ##
   ## Current Information Loss:
   ##   - IL1: 0.08
   ##   - Difference of Eigenvalues: 0.020
   ## ------------------------------------------------------------------------

## try another method
sdc <- undolast(sdc)
sdc <- addNoise(sdc, method="correlated2")
print(sdc, "numrisk")

   ## Numerical key variables: expend, income, savings
   ##
   ## Disclosure risk is currently between [0.00
   ##
   ## Current Information Loss:
   ##   - IL1: 0.11
   ##   - Difference of Eigenvalues: 0.260
   ## ------------------------------------------------------------------------
```

Exercises:

Question 3.6 **Risk for continuous variables**
Take the `testdata` data set from package **sdcMicro** again. Assume the following disclosure scenario: Set variables `expend`, `income` and `savings` as continuous key variables and optionally define a few variables as categorical key variables. Use the package **sdcMicro** to create an object of class *sdcMicroObj* by also considering the sampling weights (function argument `weightVar` in function `createSdcObj`). Apply additive noise and correlated noise and determine which of the two methods includes higher risk. Both methods can be specified in function `addNoise`.

3.12 Special Treatment of Outliers When Calculating Disclosure Risks

It is worth to show alternatives to the previous distance-based risk measure. Such alternatives took either distances between every observation into account or are based on covariance estimation (as shown here). Thus, they are computational more intensive, which is also the reason why they are not available in **sdcMicroGUI** but only in **sdcMicro** for experienced users.

Almost all data sets used in official statistics contain units whose values in at least one variable are quite different from the general observations. As a result,

these variables are very asymmetrically distributed. Examples of such outliers might be enterprises with a very high value for turnover or persons with extremely high income. In addition, multivariate outliers exist (see Templ and Meindl 2008).

Unfortunately, intruders may want to disclose a large enterprise or an enterprise with specific characteristics. Since enterprises are often sampled with certainty or have a sampling weight close to 1, 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 an observation into account. For details, see Templ and Meindl (2008). 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.

This method for risk estimation (called RMDID2 in Fig. 3.4) is also included in the **sdcMicro** package. It works as described in Templ and Meindl (2008) and is listed as follows:

1. Robust Mahalanobis distances (*RMD*) (see, for example Maronna et al. 2006) are estimated between observations (continuous variables) to obtain a robust, multivariate distance for each unit.
2. Intervals are estimated for each observation around every data point of the original data points. The length of the intervals depends on squared distances calculated in step 1 and an additional scale parameter. The higher the *RMD* of an observation, the larger the corresponding intervals.
3. Check whether the corresponding masked values of a unit fall into the intervals around the original values. If the masked value lies within such an interval, the entire observation is considered unsafe. We obtain a vector indicating which observations are safe or which are not. For all unsafe units, at least m other observations from the masked data should be very close. Close is quantified by specifying a parameter for the length of the intervals around this observation using Euclidean distances. If more than m points lie within these small intervals, we can conclude that the observation is *safe*.

Figure 3.4 depicts the idea of weighting disclosure risk intervals. For simple methods (top left and right graphics), the rectangular regions around each value are the same size for each observation. Our proposed methods take the *RMD*s of each observation into account. The difference between the bottom right and left graphics is that, for method *RMDID2*, rectangular regions are calculated around each masked variable as well. If an observation of the masked variable falls into an interval around the original value, check whether this observation has close neighbors. If the values of at least m other masked observations can be found inside a second interval around this masked observation, these observations are considered *safe*.

These methods are also implemented and available in sdcMicro as `dRisk()` and `dRiskRMD()`. The former is automatically applied to objects of class *sdcMicroObj*, while the latter has to be specified explicitly and can currently not be applied using the graphical user interface.

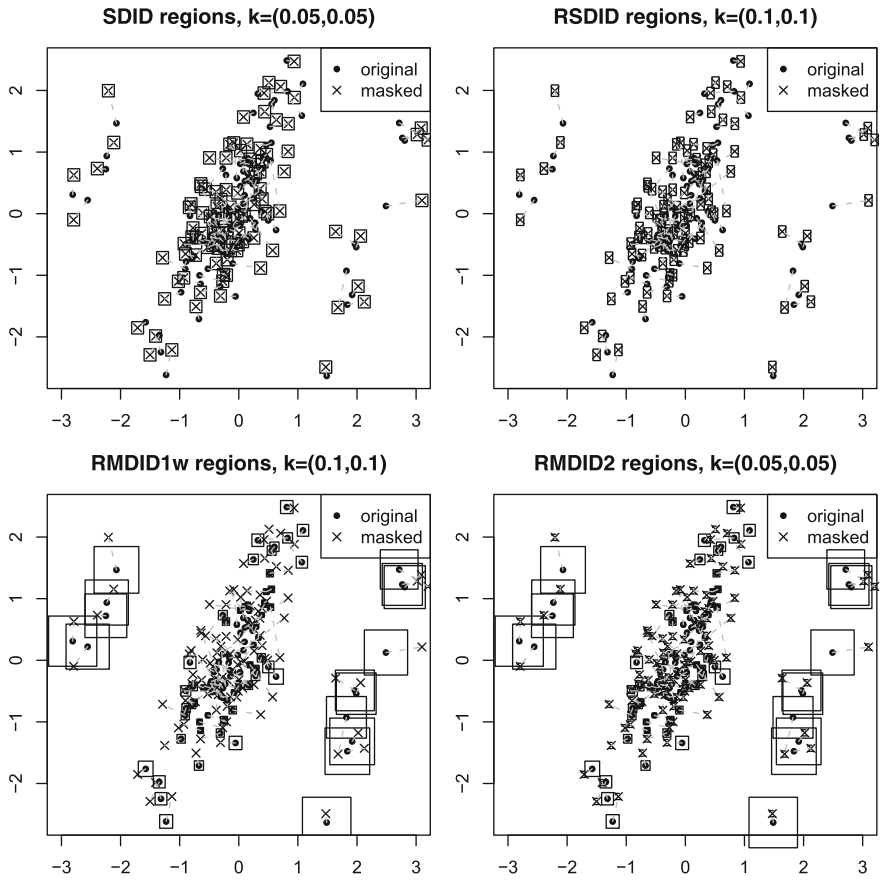


Fig. 3.4 Original and corresponding masked observations (perturbed by adding additive noise). In the *bottom right* graphic, small additional regions are plotted around the masked values for *RMDID2* procedures. The larger the intervals the more the observations is an outlier for the latter two methods. Originally published in Templ and Meindl (2008). Published with kind permission of ©Springer-Verlag Berlin Heidelberg 2008. All Rights Reserved

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

```
sdc <- dRiskRMD(sdc)
```

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

References

Agresti, A. (2002). *Categorical data analysis* (2nd ed.). Wiley Series in Probability and Statistics Hoboken: Wiley-Interscience.

Benedetti, R., & Franconi, L. (1998). Statistical and technological solutions for controlled data dissemination. In *Pre-proceedings of New Techniques and Technologies for Statistics* (pp. 225–232).

Bethlehem, J. G., Keller, W. J., & Pannekoek, J. (1990). Disclosure control of microdata. *Journal of the American Statistical Association*, *85*(409), 38–45.

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

Carlson, M. (2002a). Assessing microdata disclosure risk using the poisson-inverse gaussian distribution. *Statistics in Transition*, *5*, 901–925.

Carlson, M. (2002b). An empirical comparison of some methods for disclosure risk assessment. Technical Report, Stockholms Universitet.

Clogg, C. C., & Eliason S. R. (1987) Some common problems in log-linear analysis. *Sociological Methods & Research*, 8–44.

Elliot, M., & Manning, A. M. (2003). *Using dis to modify the classification of special uniques*. Luxembourg: In Joint UNECE/Eurostat work session on statistical data confidentiality.

Elliott, C., Skinner, C. J., & Dale, A. (1998). Special uniques, random uniques, and sticky populations: Some counterintuitive effects of geographical detail on disclosure risk. *Statistics: Research in Official*, 53–67.

Elliott, C., Manning, A. M., & Ford, R. W. (2002). A computational algorithm for handling the special uniques problem. *International Journal of Uncertainty, Fuzziness and Knowledge Based System*, *1*(1), 493–509.

Forster, J., & Webb, E. (2007). Bayesian disclosure risk assessment: Predicting small frequencies in contingency tables. Journal of the Royal Statistical Society. *Series C (Applied Statistics)*, *56*, 551–570.

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.

Horvitz, D. G., & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, *47*(260), 663–685.

Hoshino, N., & Takemura, A. (1998). On the relation between logarithmic series model and other superpopulation models useful for microdata disclosure risk assessment.

Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E. S., Spicer, K., et al. (2012). *Statistical disclosure control*. Wiley Series in Survey Methodology: Wiley. ISBN 9781118348222. https://books.google.at/books?id=BGa3zKkFm9oC.

Ichim, D. (2008). Extensions of the re-identification risk measures based on log-linear models. In J. Domingo-Ferrer & Y. Saygin (Eds.), *Privacy in statistical databases*. LNCS (vol. 5262, pp. 203–212). Heidelberg: Springer.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, *84*, 414–420.

Kowarik, A., Templ, M., Meindl, B., & Fonteneau, F. (2013). *sdcMicroGUI: Graphical user interface for package sdcMicro*. R package version 1.1.1. http://CRAN.R-project.org/package=sdcMicroGUI.

Machanavajjhala, A., Kifer, D., Gehrke, J., & Venkitasubramaniam, M. (2007). *l*-diversity: Privacy beyond *k*-anonymity. *ACM Transactions on Knowledge Discovery from Data*, *1*(1). ISSN 1556-4681.10.1145/1217299.1217302.

Manning, A. M., Haglin, D. J., & Keane J. A. (2008). A recursive search algorithm for statistical disclosure assessment. *Data Mining and Knowledge Discovery*, *16*(2), 165–196. ISSN 1384-5810.10.1007/s10618-007-0078-6. http://dx.doi.org/10.1007/s10618-007-0078-6.

Maronna, R. A., Martin, R. D., & Yohai, V. J. (2006). *Robust statistics: Theory and methods*. New York: Wiley.

Mateo-Sanz, J. M., Sebe, F., & Domingo-Ferrer, J. (2004). Outlier protection in continuous micro-data masking. *Privacy in statistical databases*. Lecture Notes in Computer Science (Vol. 3050, pp. 201–215). Springer.

Pagliuca, D., & Seri, G. (1999). Some results of individual ranking method on the system of enterprise accounts annual survey. Technical report, Esprit SDC Project, Boston. Deliverable MI-3/D2.

Polletini, S., & Seri, G. (2004). Guidelines for the protection of social micro-data using individual risk methodology. Deliverable no. 1.2-d3, CASC Project. http://neon.vb.cbs.nl/casc/.

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.

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.

Samarati, P. (2001). Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, *13*(6), 1010–1027.

Skinner, C. J., & Holmes, D. J. (1998). Estimating the re-identification risk per record in microdata. *Journal of Official Statistics*, *14*, 361–372.

Skinner, C. J., & Shlomo, N. (2006). Assessing identification risk in survey microdata using log-linear models. S3ri methodology working papers, m06/14, University of Southampton, Southampton Statistical Sciences Research Institute.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(5), 557–570.

Templ, M., Meindl, B. (2008). Robust statistics meets SDC: New disclosure risk measures for continuous microdata masking. In *Privacy in Statistical Databases*. Lecture Notes in Computer Science (Vol. 5262, pp. 177–189). Springer.

Templ, M., Meindl, B., & Kowarik, A. (2015). Statistical disclosure control for micro-data using the R package sdcMicro. *Journal of Statistical Software*, *67*(1), 1–37.

Totter, M. (2015). Disclosure risk estimation for survey data. Master's thesis, Vienna University of Technology, Vienna, Austria. supervisor: M. Templ.

Willenborg, L., & De Waal, T. (2000). *Elements of statistical disclosure control*. ISBN: 0387951210.

Chapter 4
Methods for Data Perturbation

Abstract Methods for perturbation of data differ for categorical and continuous variables. The risk for categorical key variables is dependent on the frequency counts of keys, whereas keys with only few observations are problematic. Categories of categorical key variables with low frequency counts are therefore often recoded and combined with other categories. However, a still too high disclosure risk may be present for some individuals. Local suppression is one method to further reduce the disclosure risk. In order to find a well-balanced, suitable solution, global recoding is usually applied in an explorative manner to observe with which (reasonable) recodings one achieves the best effect in terms of reducing the disclosure risk and providing high data utility. Especially with a large amount of key variables, swapping methods, such as PRAM, are good alternatives. Methods for continuous scaled variables are combining values (microaggregation) or adding noise to the values. Advanced methods such as shuffling allow to preserve certain statistics.

4.1 Kind of Methods

The SDC techniques may be categorized into three kind of methods:

- non-perturbative techniques, such as recoding and local suppression, which suppress or reduce the detail without altering the original data;
- perturbative techniques, such as adding noise, Post-Randomization Method (PRAM), micro-aggregation and shuffling, which distort the original micro-data set before release;
- techniques that generate a synthetic microdata file that preserves certain statistics or relationships of the original files.

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_4

This chapter focuses on the non-perturbative and perturbative techniques. As with disclosure risk measures, different SDC methods are applicable to categorical variables and continuous variables.

Note that generating synthetic data is a more complicated approach and the methods differ completely. This is the reason why these methods are not discussed in this chapter, and we refer to Chap. 6 that is dedicated to synthetic data simulation.

4.2 Methods for Categorical Key Variables

4.2.1 Recoding

Global recoding is a non-perturbative method that can be applied to both categorical and continuous key variables. For a categorical variable, the idea of recoding is to combine several categories into fewer categories with higher frequency counts and less detailed information. In other words, a global recoding achieves anonymity by mapping the values of the categorical key variables to generalized or altered categories. For example, one could combine multiple levels of schooling (e.g., secondary, tertiary, postgraduate) into one (e.g., secondary and above). For a continuous variable, recoding means to discretize the variable; for example, recoding a continuous income variable into a categorical variable of income levels. In both cases, the goal is to reduce the total number of possible values of a variable. Typically, recoding is applied to categorical variables to collapse categories with few observations into a single category with larger frequency counts. For example, if there are only two respondents with tertiary level of education, the tertiary can be combined with the secondary level into a single category of "secondary and above".

Remember, we have the following variables in our `sdcMicroObj`

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

The categorical key-variable `age` selected before has a lot of categories

```
table(testdata$age)
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
##  98  90 134 112 128 133 136 125 144 126 151 127 108 143 103
##  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29
## 111 106 101  96  64  88  61  64  55  55  69  56  68  69  50
##  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44
##  90  51  72  49  59  86  61  68  51  42  67  43  44  49  40
##  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59
##  65  37  31  43  28  44  28  28   8  19  31  28  17  27  20
##  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74
##  36  24  28  14  12  40   8  16  14   8  20   6  14   4   6
##  75  76  77  78  79  80  82  83  84  85  88  90  95
##   6   4   5   3   2   5   1   1   1   1   1   2   1
```

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.

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

```
## Infos on 2/3-Anonymity:
##
## Number of observations violating
## - 2-anonymity: 111 (2.424%) | in original data: 653 (14.258%)
## - 3-anonymity: 184 (4.017%) | in original data: 1087 (23.734%)
## - 5-anonymity: 345 (7.533%) | in original data: 1781 (38.886%)
##
##
------------------------------------------------------------------------
```

Let's have a look at the frequency counts of the categorized age variable. Note that we use function `extractManipData()` to extract the current data from the *sdcMicroObj* object.

```
table(extractManipData(sdc)$age)
```

```
##
##    1-9  10-19  20-29  30-39  40-49  50-59  60-69  70-79
##   1128   1110    635    629    447    250    200     70
## 80-130
##     13
```

Still some categories have only few cases, so we further modify the age variable by joining the last two groups. Here a general function could be applied, called `groupAndRename()`.

```
sdc <- groupAndRename(sdc, var = "age",
                      before = c("60-69", "70-130"),
                      after = "60-130")
print(sdc)

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 85 (1.856%) | in original data: 653 (14.258%)
   ##   - 3-anonymity: 162 (3.537%) | in original data: 1087 (23.734%)
   ##   - 5-anonymity: 294 (6.419%) | in original data: 1781 (38.886%)
   ##
   ## ----------------------------------------------------------------------

table(extractManipData(sdc)$age)

   ##
   ##    1-9  10-19  20-29  30-39  40-49  50-59 60-130
   ##   1128   1110    635    629    447    250    283
```

Alternatively we could also re-run the code using function `undolast()` and performing `globalRecode()` again with new breaks.

A special case of global recoding is top and bottom coding. This method 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. Top coding sets an upper limit on all values of a variable and replaces any value greater than this limit by the upper limit; for example, top coding would replace the age value for any individual aged above 80 with 80. Similarly, bottom coding replaces any value below a pre-specified lower limit by the lower limit; for example, bottom coding would replace the age value for any individual aged under 5 with 5. Instead of 80 or 5, respectively, one can also choose any number, e.g. the arithmetic mean of the high (or low) scores so that the arithmetic mean is not changed after top and bottom coding. As already seen in previous code blocks, in **sdcMicro** function `globalRecode()` can be used to perform both global recoding and the top/bottom coding, but there is also a specialized function called `topBotCoding` available. A help file with more examples as seen above is accessible using `?globalRecode` and `?topBotCoding`.

To replace high incomes in our test data set with the mean of the high incomes we can use the following code. Again the *sdcMicroObj* `sdc` updates its slots (e.g. risk and utility) after applying an anonymization method.

```
highIncomes <- testdata$income[testdata$income > 100000000]
sdc <- topBotCoding(sdc,
                    value = 100000000,
                    replacement = mean(highIncomes),
                    column = "income")
```

We note that **sdcMicroGUI** and the forthcoming browser-based version of the GUI offers a more user-friendly way of applying global recoding in general.

Exercises:

Question 4.1 **Global recoding**:

Take the `eusilc` data set from package **laeken** again. Assume the following disclosure scenario that defines `age`, `pb220a` (citizenship), `pl030` (education level), `rb090` (gender) and `hsize` (household size) as categorical key variables. Use the package **sdcMicro** to create an object of class *sdcMicroObj* considering the sampling weights (function argument `weightVar` in `createSdcObj`) and the household ID (function argument `hhId`). Reduce the disclosure risk through some reasonable recodings. Apply the recoding carefully, i.e. the analytical quality of the data should mostly remain the same while the disclosure risk should lower considerably.

4.2.2 Local Suppression

If unique combinations of categorical key variables remain after recoding, local suppression could be applied to the data with the aim to achieve k-anonymity. For the detailed explanation of k-anonymity, have a look in Sect. 3.3. Local suppression is a non-perturbative method typically applied to categorical variables. In this approach, missing values are created to replace certain values of individual variables to increase the number of key variables sharing the same pattern, thus reducing the record-level disclosure risks. There are two approaches to implementing local suppression. One approach sets the parameter k and tries to achieve k-anonymity (typically 3-anonymity) with minimum suppression of values. For example, in sdcMicroGUI (Kowarik et al. 2013), the user sets the value for k and orders key variables by the likelihood they will be suppressed. Then the application calls a heuristic algorithm to suppress a minimum number of values in the key variables to achieve k-anonymity. The second approach sets a record-level risk threshold. This method involves examining unsafe records with individual disclosure risks higher than the threshold and suppressing values of the selected key variable(s) for all the unsafe records.

Local suppression can be done univariate such as in Sect. 4.2.2.2, but in general it is a multivariate problem that is NP-hard, not solvable optimally in a reasonable time.

Say we have a problem P. It consists of achieving k-anonymity with the constraint to suppress as few values. In addition, a weighting determining the importance of key variables may just involve another additional parameter that can be incorporated in the constraints.

Some algorithms has been written which provide heuristic solutions.

Mondrian one solution is the algorithm Mondrian (LeFevre et al. 2006). This algorithm tries to combine categories to achieve k-anonymity, i.e. it is a sort of recoding based on the median counts of categories. However, this is a too over-simplistic approach of sorting and splitting, we do not obtained very promising results. More theoretical descriptions on the Mondrian algorithm can be found in Sect. 4.2.2.6.

all-M approach in μ-Argus and **sdcMicro** an algorithm is implemented that is often refered as the *all-M* approach. Using this algorithm, k-anonymity cannot be provided, but k-anonymity in all subsets of size M of the key variables only (in μ-Argus with the maximum of $M = 4$ variables). More precisely, the algorithm will provide k-anonymity for each combination of M key variables.

k-anonymity approach the default approach of **sdcMicro** in function `kAnon` ensures k-anonymity for the combination of all selected key variables. Naturally, this lead to k-anonymity but also to oversuppressions whenever the amount of key variables is too high, say larger than 7–10 key variables. Then often it is suitable to provide k-anonymity on a subset of variables (e.g. 7 variables) and apply a swapping method such as PRAM to the other key variables. Practical applications are shown in Sect. 4.2.2.3.

The real problem is the computation time and to find a good heuristic algorithm that solves our problem P to suppress as few values. Computation time increases a lot as soon as missing values are present in the data, since one has to treat them adequately. To decrease computation time, a lot of tricks must be used, based on (1) filtering the data in advance (2) ordering the data and (3) using C++ code and (4) thinking of good heuristics to find a good local optimal solution for this multivariate problem.

4.2.2.1 Illustrative Examples for Frequency Calculation with Missing Values

Here we show the whole problem of sample frequency calculation in case of missing values in the key variables, achieving k-anonymity and local suppression.

Remember, in Sect. 3.2.2 five possible methods how to calculate frequencies have been noted. For readability these five methods are defined once more:

1. (default method) Missing values increase frequencies in other categories.
2. (conservative method) Misssing values do not increase frequencies in other categories but in those observations where a missing occurs.
3. (category size) Missing values do increase frequencies in other categories by a factor c.
4. (conservative method 2) Misssing values do not increase frequencies in other categories.
5. (own category) Same as method 4, but missings are treated like an own category.

Table 4.1 Simple toy data set to explain different methods for sample frequency counts including missings/local suppressions in the following tables. Different solutions are displayed that show how to achieve 2-anonymity with local suppression based on different methods to calculate sample frequency counts

Original data					Method 1 (default)				
ID	Region	Status	Age group	f_k	ID	Region	Status	Age group	f_k
1	A	Single	30–49	2	1	A	Single	30–49	3
2	A	Married	30–49	2	2	A	Married	30–49	3
3	A	Married	30–49	2	3	A	Married	30–49	3
4	A	Single	30–49	2	4	A	Single	30–49	3
5	A	Widow	30–49	1	5	A	*	30–49	5
Method 2 (conservative)					Method 3 (category size)				
ID	Region	Status	Age group	f_k	ID	Region	Status	Age group	f_k
1	A	Single	30–49	2	1	A	Single	30–49	2.4
2	A	Married	30–49	2	2	A	Married	30–49	2.4
3	A	Married	30–49	2	3	A	Married	30–49	2.4
4	A	Single	30–49	2	4	A	Single	30–49	2.4
5	A	*	30–49	5	5	A	*	30–49	5 (or 3.2)
					2.4 = ratio single · 1 missing 2.4 = ratio married · 1 missing				
Method 4 (conservative 2)					Method 5 (own category)				
ID	Region	Status	Age group	f_k	ID	Region	Status	Age group	f_k
1	A	*	30–49	5	1	A	*	30–49	3
2	A	Married	30–49	2	2	A	Married	30–49	2
3	A	Married	30–49	2	3	A	Married	30–49	2
4	A	*	30–49	5	4	A	*	30–49	3
5	A	*	30–49	5	5	A	*	30–49	3

To motivate and explain these different concepts, a toy data example is used in the following, see Table 4.1 (upper left), with different values only in the second variable. It can be seen that 2-anonymity is not reached in the original data. For our scenarios, we assume that 2-anonymity should be achieved by local suppressions, i.e. the necessary suppressions under each frequency count method are made and sample frequencies are calculated. The results are displayed in Table 4.1. The default method, the conservative method and the category size method lead to the lowest numbers of necessary suppression to achieve 2-anonymity, i.e. with one suppression

Table 4.2 Simple toy data set to explain different methods for sample frequency counts including missings/local suppressions in the following tables. Different solutions are displayed to achieve 3-anonymity with local suppression based on different methods to calculate sample frequency counts

Original data					Method 1 (default)				
ID	Region	Status	Age group	f_k	ID	Region	Status	Age group	f_k
1	A	Single	30–49	2	1	A	Single	30–49	3
2	A	Married	30–49	2	2	A	Married	30–49	3
3	A	Married	30–49	2	3	A	Married	30–49	3
4	A	Single	30–49	2	4	A	Single	30–49	3
5	A	Widow	30–49	1	5	A	*	30–49	5
Method 2 (conservative)					Method 3 (category size)				
ID	Region	Status	Age group	f_k	ID	Region	Status	Age group	f_k
1	A	*	30–49	5	1	A	*	30–49	5
2	A	*	30–49	5	2	A	Married	30–49	3.2
3	A	*	30–49	5	3	A	Married	30–49	3.2
4	A	*	30–49	5	4	A	*	30–49	5
5	A	*	30–49	5	5	A	*	30–49	5
					$3.2 = 2 +$ ratio of married $\cdot$ 3 missings				
Method 4 (conservative 2)					Method 5 (own category)				
ID	Region	Status	Age group	f_k	ID	Region	Status	Age group	f_k
1	A	*	30–49	5	1	A	*	30–49	5
2	A	*	30–49	5	2	A	*	30–49	5
3	A	*	30–49	5	3	A	*	30–49	5
4	A	*	30–49	5	4	A	*	30–49	5
5	A	*	30–49	5	5	A	*	30–49	5

2-anonymity is reached. However, the sample frequencies are different. The method using an own category for the missing values is most strict. It leads to the lowest frequency counts with three suppressions.

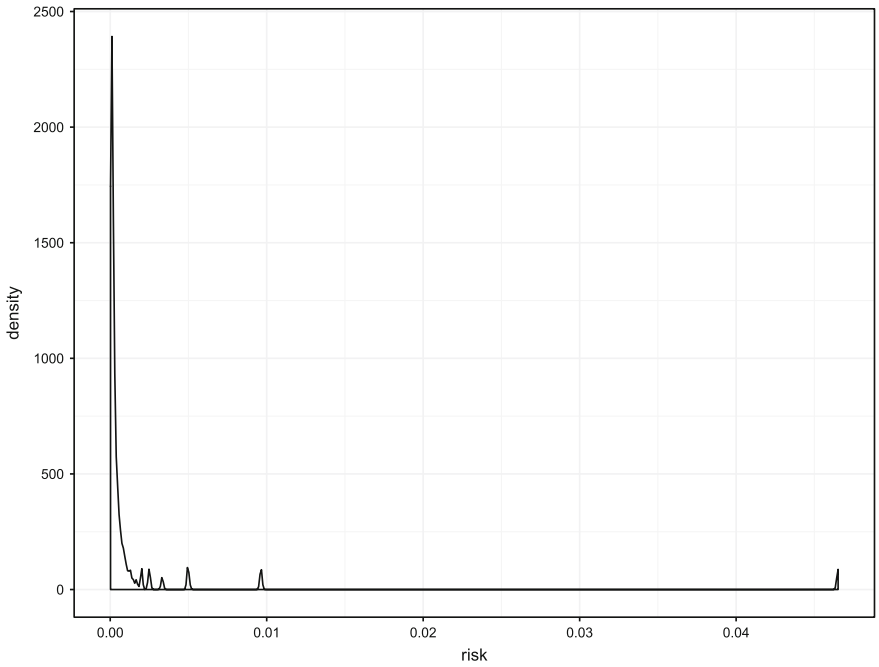
As from the previous table, from the upper left table in Table 4.2 it can be seen that 3-anonymity is not reached. For our scenarios, we assume that this time 3-anonymity should be achieved by local suppressions. The results are displayed in Table 4.2. The default method count a missing as a any possible category. Thus one suppression is already enough to achieve 3-anonymity. The category size methods counts the amount of values of category *married* plus the ratio of category *married* to the amount of categories times the number of missing values. All other approaches are more conservative and all values of *status* have to be suppressed to achieve 3-anonymity.

The choice of method for sample frequency calculation is best done with discussion with the law department or the management of the data holder. In the following, the so called default method is used. This method is used for years at Statistics Austria and every organisation who uses the package **sdcMicro**. The default method is amongst possible choices the less conservative one. It assumes that an intruder cannot be sure about the correct category of the suppressed value.

4.2.2.2 Univariate Local Suppression for Observations with High Risk

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 a threshold. This is illustrated in the following code listing. First, a density plot of the individual risks is plotted. This plot helps to determine the threshold. The majority of the data have very low risk, below 0.005, which already may serve as the value of the threshold. For those observations with higher individual risk than 0.005, the values of variable “urbur” are suppressed.

```
## extract vector of individual risks
risk <- get.sdcMicroObj(sdc, "risk")$individual[, "risk"]
## have a look on the risk
ggplot(data.frame(risk), aes(x=risk)) + geom_density() + theme_bw()
```




```
## suppress values in urbrur for risk higher than 0.05
sdc <- localSupp(sdc, keyVar = "urbrur", threshold = 0.05)
```

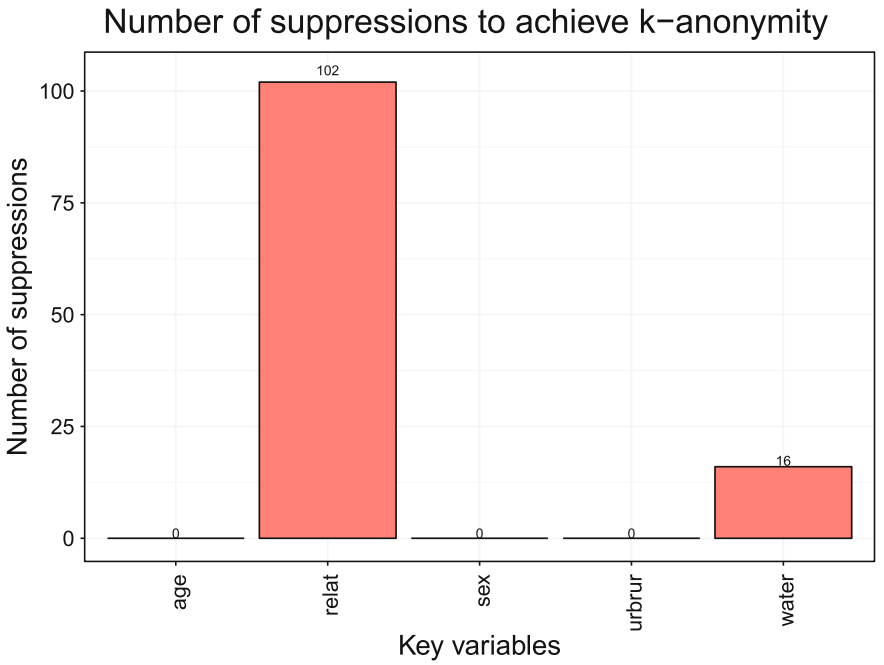
4.2.2.3 Ensuring *k*-Anonymity—The Optimal Approach

To automatically suppress a minimum amount of values in the key variables to achieve *k*-anonymity, one can use function `kAnon()`. In the code listing below, first the previous local suppression is reverted. Local suppression to achieve 3-anonymity is then done by setting parameter `k = 3`.

```
## undo last step (localSupp(...))
sdc <- undolast(sdc)

## local suppression to ensure 3-anonymity
## for given key variables
sdc <- kAnon(sdc, k=3)
```

The amount of suppression can be reported via a plot of the object `sdc`.

```
plot(sdc, "ls")
```



Also the print of this object gives information about the number and percentages of suppressions for each variable.

```
print(sdc, "ls")

   ## Local suppression:
   ##  KeyVar | Suppressions (#) | Suppressions (%)
   ##  urbrur |                 0 |            0.000
   ##   water |                16 |            0.349
   ##     sex |                 0 |            0.000
   ##     age |                 0 |            0.000
   ##   relat |               102 |            2.227
   ## ---------------------------------------------------------------------
```

In this implementation, a heuristic algorithm is called to suppress as few values as possible. We see that 3-anonymity is ensured. In brackets the number of observations violating 2- and 3-anonymity in the original data set are reported.

```
print(sdc, "kAnon")

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 0 (0.000%) | in original data: 653 (14.258%)
   ##   - 3-anonymity: 0 (0.000%) | in original data: 1087 (23.734%)
   ##   - 5-anonymity: 138 (3.013%) | in original data: 1781 (38.886%)
   ##
   ## ----------------------------------------------------------------------
```

Importance of variables: It is possible to specify a desired ordering of key variables. If you have a look at the functions arguments in `kAnon()`, it can be seen that the function argument `importance` is responsible for it. The aim is that the higher the importance of a variable, the less suppressions are taken for this variable.

In other words, the algorithm allows the specification of a preference-vector that determines which key variables should be preferred when selecting values to be suppressed. By specifying this importance of variables as a parameter in `kAnon()`, for key variables with high importance, suppression will only take place if no other choices are possible. Still, it is possible to achieve k-anonymity for selected key variables.

```
## undo last local suppression
sdc <- undolast(sdc)
```

Remember, the key variables are

```
str(testdata[, sdc@keyVars])

   ## 'data.frame':    4580 obs. of  5 variables:
   ##  $ urbrur: int  2 2 2 2 2 2 2 2 2 2 ...
   ##  $ water : int  3 3 3 3 3 3 3 3 3 3 ...
   ##  $ sex   : int  1 2 1 1 1 2 2 2 1 2 ...
   ##  $ age   : int  46 41 9 6 52 47 13 19 9 16 ...
   ##  $ relat : int  1 2 3 3 1 2 3 3 3 3 ...
```

Another importance is assigned to those variables. In general, the lower the number, the less local suppressions. For any reason, we assume that variable *relat* is very important, giving the importance of 1. Variable *age* seems more important than *urbrur*, *water* and *sex*, thus giving age importance of 2. The rest of the variables are assigned by the numbers 5 (*urbrur*), 4 *water* and 3 *sex*. The result is now different from before.

```
sdc <- kAnon(sdc, importance = c(5,4,3,2,1), k = 3)
print(sdc, "ls")
```

```
## Local suppression:
##  KeyVar | Suppressions (#) | Suppressions (%)
##  urbrur |               44 |            0.961
##   water |               52 |            1.135
##     sex |               13 |            0.284
##     age |                9 |            0.197
##   relat |                4 |            0.087
## ----------------------------------------------------------------------
```

It can be seen that now other variables are mainly used for suppressions. In variable *relat*, for example, only 3 instead of 114 local suppression are made.

Stratification:

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

4.2.2.4 An Approach for Ensuring k-anonymity in Subsets

Due to computational issues another approach exists in μ-Argus, often refered as all-M approach. It is also implemented in **sdcMicro** (with no computational constraints), but mainly for comparison reasons. With this approach k-anonymity is usually not reached considering all key variables, but it can ensure k-anonymity on subsets of key variables.

It is also possible to provide k-anonymity for subsets of key-variables with varying parameter k. In the follow-up example we want to provide 10-anonymity for all combinations of 4 key variables, 20-anonymity for all combinations with 3 key variables and 30-anonymity for all combinations of 2 key variables. Note that strata are automatically considered.

```
sdc <- undolast(sdc) combs
<- 4:2 k <-
c(5,10,20)
sdc <- kAnon(sdc,
k=k, combs=combs)
print(sdc, "ls")

   ## Local suppression:
   ##  KeyVar | Suppressions (#) | Suppressions (%)
   ##  urbrur |                 0 |            0.000 ##   water |
   96 |            2.096 ##     sex |                0 |
   0.000 ##     age |               98 |            2.140 ##   relat |
   147 |            3.210 ##
   ----------------------------------------------------------------------
```

```
print(sdc, "kAnon")
```

```
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##   - 2-anonymity: 0 (0.000%) | in original data: 653 (14.258%)
##   - 3-anonymity: 3 (0.066%) | in original data: 1087 (23.734%)
##   - 5-anonymity: 30 (0.655%) | in original data: 1781 (38.886%)
##
## ----------------------------------------------------------------------
```

Since both, k-anonymity is not ensured by this approach and a lot of parameters have to be set (k's and combinations), we strongly suggest to use the approach from the previous Sect. 4.2.2.3. In addition, more suppressions are made using this approach based on subsets.

However, if for any reason the SDC specialist decides to have a large amount of key variables this approach becomes important.

4.2.2.5 Choice of the Local Suppression Algorithm

The "Argus" subset approach and the optimal approach are equal if `combs = #` number of key variables. In any other case, the optimal approach ensures k-anonymity while the subset approach cannot ensure this. In addition, several parameters have to be determined when using the subset approach.

In general, it is advisable to provide k-anonymity for the most important indirect identifiers, i.e. to apply the optimal approach. The number of key variables is usually a number between 4 and 8. Scenarios with more than 8 key variables are seldom and rarely reported in literature. The optimal approach is preferable in this case since it ensures k-anonymity and does not lead to much over-suppression.

With a very high number of key variables, say 15, the optimal approach will reach k-anonymity but, because the high number of possible keys, many values might be suppressed.

With such a large number of key variables, one alternative is to apply PRAM (see Sect. 4.2.3) on some (e.g. 7 variables) and the optimal local suppression approach guarantees k-anonymity for the remaining variables (e.g. 8 variables). However, since it is not easy to determine the disclosure risk for prammed variables, alternatives are suggested in the following.

If PRAM should not be applied, k anonymity should be ensured in subsets of variables. Ideally, the number of variables in a subset should be high, e.g. 8. Then, 6435 combinations of 8 out of 15 variables exist. Thus, k-anonymity is ensured in each of the 6435 combinations by using the subset local suppression approach.

In the previous section, it can be seen that three values of k for three combinations are given. But using `combs <- 4:2` and `k <- c(5, 10, 20)` was somehow a subjective decision. On the one hand, one wants to guarantee low disclosure risk. With this aim, the values for `combs` should be high as well as the corresponding values of k. On the other hand, one wants to result in a low number of suppressions, thus low numbers of `comb` and `k` should be chosen. Currently, there is no algorithm available which would solve this optimization problem.

4.2.2.6 Mondrian

A very simple approach which is often cited is the Mondrian algorithm (LeFevre et al. 2006) which is based on multidimensional partitioning. For each split/partitioning, the key variable with the highest number of categories is chosen. The data are split according to the median in each partition whenever a split is possible. A split is possible when the splitted data contains enough observations. More precisely, this greedy (strict) median-partitioning algorithm results in a set of multidimensional regions, each containing between k and $2p(k-1)+m$, with p the number of key variables and m the number of distinct categories (for more details, see LeFevre et al. 2006).

This algorithm ensures k-anonymity but due to its simple procedure it may lead to over-suppressions. The algorithm is not included in **sdcMicro** but available upon request.

4.2.2.7 Linked Variables in Local Suppression

As mentioned in Chap. 2 after applying local suppression (`kAnon` or `localSuppression`) ghost/linked variables should have the same suppression pattern as the variable that they are linked to.

We give an illustrative artificial example to show how this concept works. Another variable with the same content is produced, in the following this is variable `electcon2`, `electcon3` (linked to `electcon` and `water2` (linked to `water`).

```
data("testdata", package = "sdcMicro")
testdata$electcon2 <- testdata$electcon
testdata$electcon3 <- testdata$electcon
testdata$water2 <- testdata$water
keyVars <- c("urbrur","roof","walls","water","electcon","relat","sex")
numVars <- c("expend","income","savings")
w <- "sampling_weight"
```

We want to make sure that some variables not used as key-variables will have the same suppression pattern as variables that have been selected as key variables. Thus, we are using *ghost*-variables. As indicated above, in our example we want variables `electcon2` and `electcon3` to be linked to key-variable `electcon` and we want variable`water2` to be linked to key-variable `water`.

```
ghostVars <- list()
ghostVars[[1]] <- list()
ghostVars[[1]][[1]] <- "electcon"
ghostVars[[1]][[2]] <- c("electcon2","electcon3")
ghostVars[[2]] <- list()
ghostVars[[2]][[1]] <- "water"
ghostVars[[2]][[2]] <- "water2"
```

Next we create the *sdcMicroObj* object.

```
obj <- createSdcObj(testdata, keyVars=keyVars,
  numVars=numVars, w=w, ghostVars=ghostVars)
```

We apply 3-anonymity to selected key variables

```
obj <- kAnon(obj, k=3)
obj
```

```
obj <- kAnon(obj, k=3)
obj
```

```
## Data set with 4580 rows and 17 columns.
##  --> Categorical key variables: urbrur, roof, walls, water, electcon,
##                                 relat, sex
##  --> Numerical key variables: expend, income, savings
##  --> Weight variable: sampling_weight
## ----------------------------------------------------------------------
##
## Information on categorical Key-Variables:
##
## Reported is the number, mean size and size of the smallest category
## for recoded variables.
## In parenthesis, the same statistics are shown for the unmodified data.
## Note: NA (missings) are counted as separate categories!
##
##  Key Variable Number of categories     Mean size
##        urbrur                   2 (2)  2290.000 (2290.000)
##          roof                   6 (5)   913.600  (916.000)
##         walls                   3 (3)  1526.667 (1526.667)
##         water                   9 (8)   569.250  (572.500)
##      electcon                   4 (3)  1525.333 (1526.667)
##         relat                   9 (9)   554.000  (508.889)
##           sex                   2 (2)  2290.000 (2290.000)
##  Size of smallest
##               646  (646)
##                15   (16)
##                50   (50)
##                25   (26)
##               103  (107)
##                 3    (1)
##              2284 (2284)
## ----------------------------------------------------------------------
##
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##  - 2-anonymity: 0 (original data: 157)
##  - 3-anonymity: 0 (original data: 281)
##
## Percentage of observations violating
##  - 2-anonymity: 0.000 % (original data: 3.428 %)
##  - 3-anonymity: 0.000 % (original data: 6.135 %)
## ----------------------------------------------------------------------
##
## Numerical key variables: expend, income, savings
##
## Disclosure risk (~100.00% in original data):
##  modified data: [0.00%; 100.00%]
##
## Current Information Loss in modified data (0.00% in original data):
##  IL1: 0.00
##  Difference of Eigenvalues: 0.000%
## ----------------------------------------------------------------------
##
## Local Suppression:
##    KeyVar | Suppressions (#) | Suppressions (%)
##    urbrur |                 0 |            0.000
##      roof |                12 |            0.262
##     walls |                 0 |            0.000
##     water |                26 |            0.568
##  electcon |                 4 |            0.087
##     relat |               148 |            3.231
##       sex |                 0 |            0.000
## ----------------------------------------------------------------------
```

In the following we can see that the suppression patterns of the key variable and its ghost variables are identical.

```
manipGhostVars <- get.sdcMicroObj(obj, "manipGhostVars")
manipKeyVars <- get.sdcMicroObj(obj, "manipKeyVars")
all(is.na(manipKeyVars$electcon) == is.na(manipGhostVars$electcon2))
```

```
## [1] TRUE
```

```
all(is.na(manipKeyVars$electcon) == is.na(manipGhostVars$electcon3))
```

```
## [1] TRUE
```

```
all(is.na(manipKeyVars$water) == is.na(manipGhostVars$water2))
```

```
## [1] TRUE
```

Exercises:

Question 4.2 **Local suppression**:
Take the `eusilc` data set from package **laeken** and the *sdcMicroObj* object produced in the previous example including all already done recodings. The task is now to ensure k-anonymity.

4.2.3 Post-randomization Method (PRAM)

If there are a larger number of categorical key variables (e.g., more than 5), recoding might not sufficiently reduce disclosure risks, or local suppression might lead to great information loss. In this case, the PRAM (Gouweleeuw et al. 1998) may be a more efficient alternative.

PRAM (Gouweleeuw et al. 1998) is a probabilistic, perturbative method for protecting categorical variables. The method swaps the categories for selected variables based on a pre-defined transition matrix, which specifies the probabilities for each category to be swapped with other categories.

To illustrate, consider the variable location, with three categories: *location* = 1 "east", *location* = 2 "middle", *location* = 3 "west". We define a 3-by-3 transition matrix, where p_{ij} is the probability of changing category i to j. For example, in the following matrix,

$$\mathbf{P} = \begin{pmatrix} 0.1 & 0.9 & 0 \\ 0.2 & 0.1 & 0.7 \\ 0.9 & 0 & 0.1 \end{pmatrix}$$

the probability that the value of the variable will stay the same after perturbation is 0.1, since we set $p_{11} = p_{22} = p_{33}$. The probability of east being changed into middle is p_{12}, while east will not be changed into west because p_{13} is set to be 0. PRAM protects the records by perturbing the original data file, while at the same time, since the probability mechanism used is known, the characteristics of the original data can be estimated from the perturbed data file. PRAM can be applied to each record independently, allowing the flexibility to specify the transition matrix as a function parameter according to desired effects. For example, it is possible to prohibit changes from one category to another by setting the corresponding probability in the transition matrix to 0, as shown in the example above. It is also possible to apply PRAM to subsets of the microdata independently.

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

```
Apramed <- pram(A); 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:

```
summary(Apramed)
```

```
## Variable:  x
##  ----------------------
## Frequencies in original and perturbed data:
##                                   x A1 A2 A3 NA
## 1:           Original Frequencies  5  5  5  0
## 2: Frequencies after Perturbation  6  5  4  0
##
## Transitions:
##    transition Frequency
## 1:    1 --> 1         5
## 2:    2 --> 2         5
## 3:    3 --> 1         1
## 4:    3 --> 3         4
```

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 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 of the data set.

We again create an object of class *sdcMicroObj*, also specifying a strata variable to apply PRAM within these strata independently. Note that when applying PRAM to variables, these variables should be saved as a `factor`.

```
require("sdcMicro")
data(testdata, package="sdcMicro")
# categorical variables should be saved as a factor
vars <- c('urbrur', 'water', 'sex', 'age', 'relat', 'walls', 'roof')
testdata[, vars] <- lapply(testdata[, vars], as.factor)

# setting up the SDC problem
sdc <- createSdcObj(testdata,
          keyVars = c('urbrur', 'water', 'sex', 'age', 'relat'),
          numVars = c('expend', 'income', 'savings'),
          pramVars = c("walls"),
          w = 'sampling_weight',
          hhId = 'ori_hid',
          strataVar = 'hhcivil'
          )
```

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

```
## Post-Randomization (PRAM):
## Variable: walls
## --> final Transition-Matrix:
##             2         3           9
## 2 0.84447887 0.1499160 0.005605102
## 3 0.05420769 0.9410131 0.004779201
## 9 0.13485876 0.3180081 0.547133184
##
## Changed observations:
##   variable nrChanges percChanges
## 1    walls       398        8.69
## ---------------------------------------------------------------------------
```

Sometimes it is useful to apply pram not on all categories. To give an illustrative example, children with age below 16 should not be prammed in variable education. Or governmental organisations should not be prammed in variable ownership.

We show an example for our testdata set. We want to apply pram to variable *urbrur* for each group of variable *urbrur*. However, no value should be changed where *roof*

equals 4. Thus we are creating a new value for these observations. First the previous application of PRAM is undone, then a new category is made for those observations that should not change.

```
sdc <- undolast(sdc)
sv <- testdata$urbrur
levels(sv) <- c("1","2","3")
sv[testdata$roof == 4] <- 3
```

Next PRAM is applied. For a later check, the original data as well as the prammed variable *roof* is extracted.

```
sdc <- pram(sdc, variables = "roof", strata_variables = sv)
orig <- get.sdcMicroObj(sdc, "origData")$roof
pramed <- get.sdcMicroObj(sdc, "manipPramVars")$roof
```

Now it can be validated if any of the category 4 in the prammed variable differs from the original category.

```
all(pramed[orig == 4] == 4)

   ## [1] FALSE
```

It can be seen that nothing is changed for this category.

4.3 Methods for Continuous Key Variables

4.3.1 Microaggregation

Micro-aggregation (Defays and Anwar 1998) is a perturbing method typically applied to continuous variables. It is also a natural approach to achieving k-anonymity. The method first partitions records into groups, then assigns an aggregate value (typically the arithmetic mean, but other robust methods are also possible) to each variable in the group. As an example, in Table 4, records are first partitioned into groups of two, and then the values are replaced by the group means. Note that in the example, by setting group size of two, micro-aggregation automatically achieves 2-anonymity with respect to the three key variables.

Individual values of the records for each variable are replaced by the group aggregation value, which is often the mean; as an example, see Table 4.3, where two values that are most similar are replaced by their column-wise means.

To preserve the multivariate structure of the data, the most challenging part of micro-aggregation is grouping records by how "similar" they are. The simplest method is to sort data based on a single variable in ascending or descending order. Another option is to cluster data first, and sort by the most influential variable in each cluster. These methods, however, might not be optimal for multivariate data (Templ and Meindl 2008).

Table 4.3 Example of micro-aggregation. Columns 1–3 contain the original variables, columns 4–6 the micro-aggregated values (rounded on two digits)

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

Individual ranking method

The individual ranking methods (Defays and Anwar 1998) is often applied because of its simplicity. The method replaces values by its aggregates column by column independently. First, the first column is sorted and the index of sorting is memorized to be able to sort the values back in the original order. Then the first k values are replaced by their aggregate (usually the arithmetic mean), the next k values are replaced by their aggregate, and so on, until all values are aggregated from the first variable. The variable is then back-sorted. This procedure is then applied on the other variables independently.

Microaggregation based on PCA

The Principle Component Analysis method sorts data on the first principal components (see, e.g.,Templ and Meindl 2008). A robust version of this method can be applied to clustered data for small- or medium-sized datasets (see, e.g.,Templ and Meindl 2008). This approach is fast and performs well whenever the first principal component explains a high percentage of the variance for the variables considered for micro-aggregation. If this is not the case, other algorithms are preferable, such as the MDAV algorithm explained next.

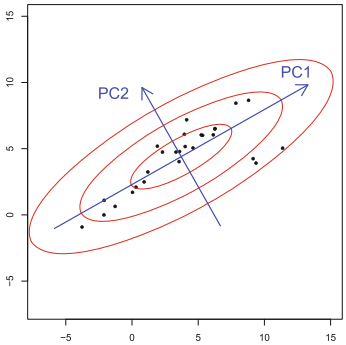
Figure 4.1 shows the procedure. First, the first principal component must be estimated and along the first principal component the values are then aggregated. As for almost any microaggregation procedure, the remaining $(2k - 1)$ are microaggregated whenever $nmodulok$ is unequal zero.

Microaggregation by MDAV

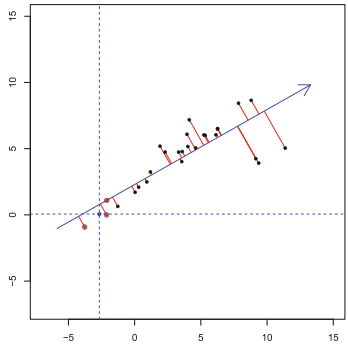
The Maximum Distance to Average Vector (MDAV) method is a standard method that groups records based on classical Euclidean distances in a multivariate space (Domingo-Ferrer and Mateo-Sanz 2002).

Figure 4.2 shows the MDAV procedure, which works as follows.

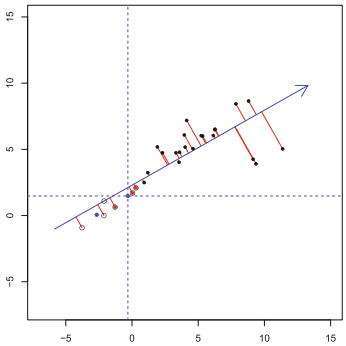
1. The center of the data is estimated using column-wise arithmetic means.
2. The farthest observation (Euclidean distance), say $\mathbf{x}_r$ from the center is then chosen.



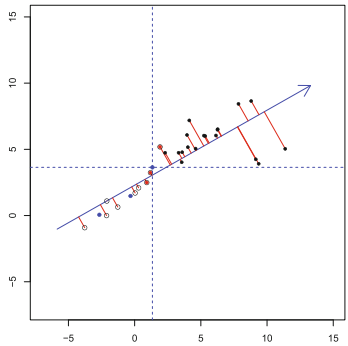
(a) First two principal components.



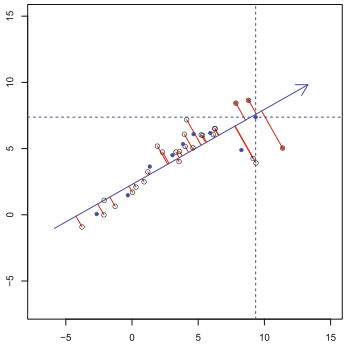
(b) Projection/sorting on first component. Aggregation along the sorted values (first three observations).



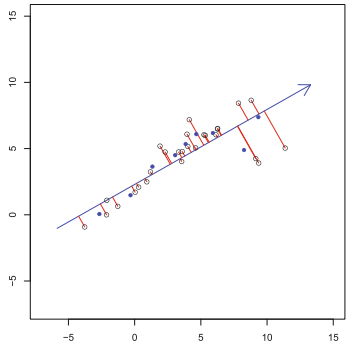
(c) Aggregation along the sorted values (next three observations).



(d) Aggregation along the sorted values (next three observations).

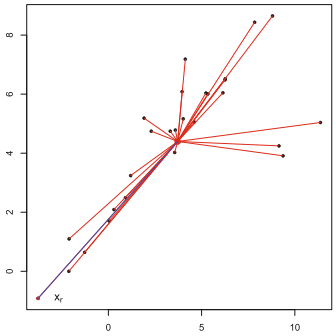


(e) Aggregation along the sorted values (remaining last observations).

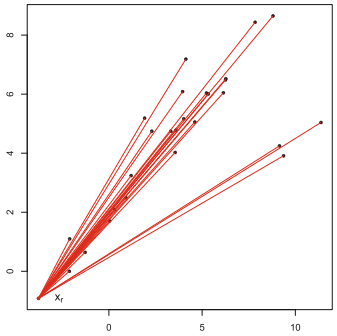


(f) Microaggregated values in blue.

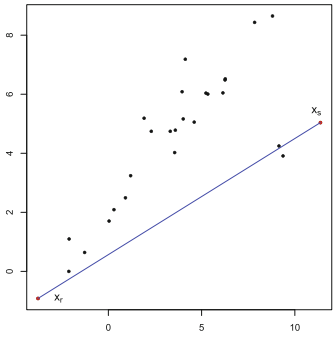
Fig. 4.1 Schematic workflow of microaggregation using principal component analysis on two-dimensional toy data. Aggregation is done along the first principal component



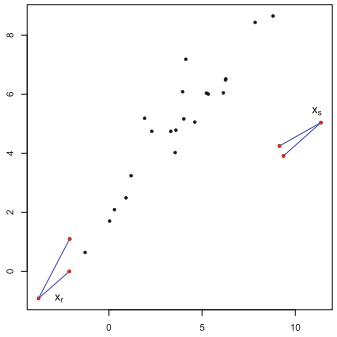
(a) Distances to the center.



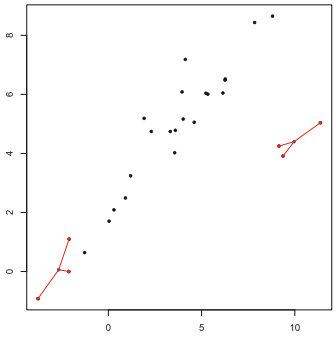
(b) Distances from observation $\mathbf{x}_r$ that is farthest from center to all other observations.



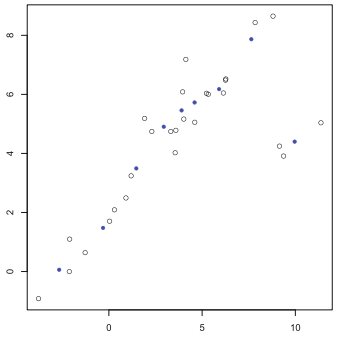
(c) Farthest point $\mathbf{x}_s$ from $\mathbf{x}_r$.



(d) Determining the $k-1$ (here: 2) nearest neighbors of $\mathbf{x}_r$ and $\mathbf{x}_s$.



(e) Replacement/aggregation with the mean.



(f) Resulting microaggregated observations (in blue) after applying the same procedure as long as all observations are microaggregated.

Fig. 4.2 Schematic workflow of microaggregation using MDAV on two-dimensional toy data. Aggregation is stepwise applied after all observations are microaggregated. The first step is shown in (**b**)–(**e**)

3. The farthest observation (Euclidean distance) from $\mathbf{x}_s$ is also selected.
4. $k-1$ nearest neighbors (using Euclidean distances) are chosen for $\mathbf{x}_r$, and also for $\mathbf{x}_s$ and the arithmetic column-wise means are estimated for both groups of observations.
5. The corresponding observations are replaced by their arithmetic column-wise means.
6. Start at (1) for the remaining observations. A variant of the procedure is to start not at step (1) but at step (2), i.e. not to estimate the centers in each step but held them fixed. This iteration is done until less than or equal $2k-1$ observations are left.
7. Aggregate the remaining observations.

Microaggregation by robust Mahalanobis distances

The MDAV method was further improved by replacing Euclidean distances with robust multivariate (Mahalanobis) distance measures (Templ and Meindl 2008). All of these methods are implemented in **sdcMicro** (Templ et al. 2015) and **sdcMicroGUI** (Kowarik et al. 2013).

Microaggregation for mixed scale data

To deal not only with continuous key variables but also with categorical (and ordered categorical or semi-continuous) in the same time, two central issues have to be changed. First, a more general definition of distance is required and secondly, for non-continuous variables a strategy for aggregation has to be developed.

For the first task, this microaggregation method is using distances computed similar to the Gower distance (Gower 1971). The distance function makes distinction between the variable types categorical, ordered categorical, continuous and semi-continuous (variables with a fixed probability mass at a constant value, e.g. 0). The distance calculation is explained in Sect. 5.2.1.

After calculating the distances and to microaggregate, it is necessary to sample a category in each group whenever a variable is non-(semi-)continuous. This probabilities corresponding to the occurrence of the level in the groups. The level with the most occurrences is then chosen, or if the maximum is not unique it is selected randomly.

Applying microaggregation in **sdcMicro** is straightforward, we again apply the corresponding function to our *sdcMicroObj* object, and the utility and risk will be automatically updated as well as other slots such as `manipData`.

```
sdc <- microaggregation(sdc)
print(sdc, "numrisk")
```

```
## Numerical key variables: expend, income, savings
##
## Disclosure risk is currently between [0.00%; 36.09%]
##
## Current Information Loss:
##   - IL1: 0.11
##   - Difference of Eigenvalues: 0.060%
## ------------------------------------------------------------------------
```

Depending on the chosen method in function `microaggregation()`, additional parameters can be specified. For example, it is possible to specify the number of observations that should be aggregated (parameter `aggr`) as well as the statistics used to calculate the aggregation (parameter `measure`). This statistics defaults is the arithmetic mean. It is also possible to perform micro-aggregation independently to pre-defined strata (the parameter is called `strata_variables`) or to use cluster methods to achieve the grouping (depending on parameter `method`). Let us undo the last action and perform microaggregation using different parameters. As mentioned before, the disclosure risk updates automatically.

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

   ## Numerical key variables: expend, income, savings
   ##
   ## Disclosure risk is currently between [0.00%; 25.35%]
   ##
   ## Current Information Loss:
   ##   - IL1: 0.13
   ##   - Difference of Eigenvalues: 0.100%
   ## ------------------------------------------------------------------------
```

We can observe that the disclosure risk decreased considerably – $\sim$ only 2.88% of the original values do not fall within intervals calculated around the perturbed values, compare Sect. 3.11 on distance-based risk estimation. We can 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 microaggregtion 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. its application is also straightforward, just apply the function to an object of class *sdcMicroObj*. Note that in such an object, every important information is already stored and the variables to be microaggregated are chosen automatically.

```
sdc <- microaggrGower(sdc)
```

4.3.2 Noise Addition

Adding noise is a perturbative method typically applied to continuous variables. The idea is to add or multiply a stochastic or randomized number to the original values to protect data from exact matching with external files. While this approach sounds simple in principle, many different algorithms can be used. In this section, we introduce the uncorrelated and correlated additive noise methods.

Adding noise should be used with caution, as the results depend greatly on the amount of noise chosen.

4.3.2.1 Uncorrelated Additive Noise

Uncorrelated additive noise (see, e.g., Hundepool et al. 2007) can be expressed as the following:

$$\mathbf{z}_j = \mathbf{x}_j + \epsilon_j \quad , \tag{4.1}$$

where vector $\mathbf{x}_j$ represents the original values of variable j, $\mathbf{z}_j$ represents the perturbed values of variable j and ϵ_j (uncorrelated noise, or white noise) denotes normally distributed errors with $Cov(\epsilon_l, \epsilon_k) = 0$ for all $k \neq l$. In matrix notation this looks like

$$\mathbf{Z} = \mathbf{X} + \epsilon \quad , \tag{4.2}$$

with $X \sim (\mu, \Sigma)$, $\epsilon \sim N(0, \Sigma_\epsilon)$ and $\Sigma_\epsilon = c \cdot diag(\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2)$, for a constant $c > 0$.

In the following code, synthetic toy data are simulated and noise is added. The original but also the perturbed data are displayed in Fig. 4.3. Noise can be added to variables using function `addNoise()` and its parameter `noise`. The corresponding help file can be accessed with `?addNoise`. In addition, a 97.5% tolerance ellipse showing the covariance structure is drawn for both the original and the perturbed data. More noise than probably needed is added to the original variables in order to show the flaws of this method. It is clearly visible that the covariance of the data set is not respected in the noise addition. The perturbed data have a different covariance structure than the original data.

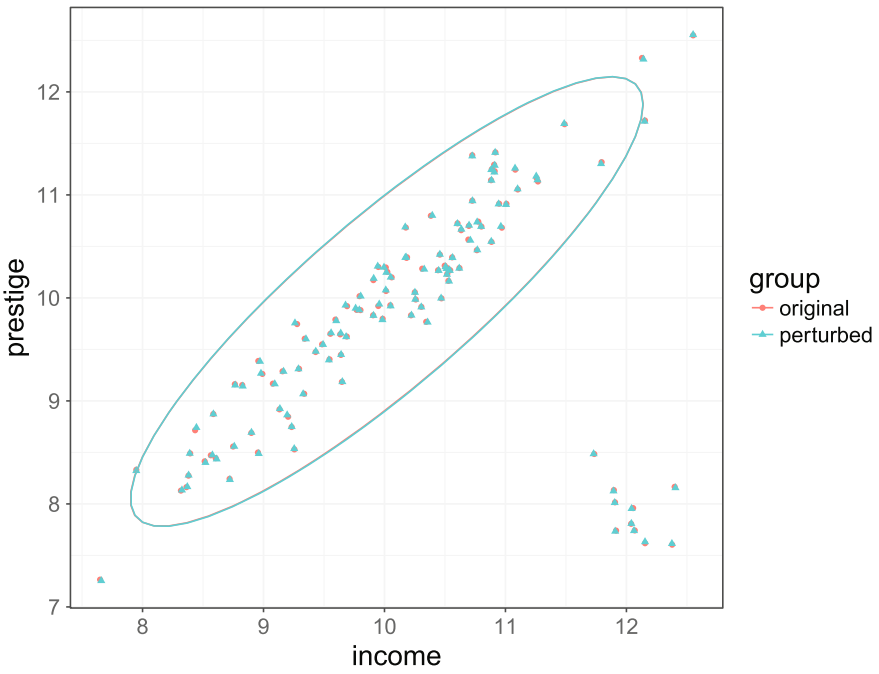


Fig. 4.3 Adding additive noise. More noise than probably needed is added to see the flaws of the method

```
## we generate some synthetic bivariate toy data to illustrate noise addition
X <- MASS::mvrnorm(100,
                   mu = c(10,10),
                   Sigma = matrix(c(1,0.95,0.95,1),
                                  byrow = TRUE, ncol=2))
X2 <- MASS::mvrnorm(10,
                    mu = c(12,8),
                    Sigma = matrix(c(0.1,0,0,0.1),
                                  byrow = TRUE, ncol=2))
X <- data.frame(rbind(X, X2))
colnames(X) <- c("income","prestige")
head(X)
```

```
##      income  prestige
## 1 10.887193 11.249357
## 2 11.101999 11.053242
## 3 12.128663 12.329934
## 4  8.585157  8.873169
## 5  8.899352  8.689755
## 6  8.564348  8.472442
```

```
## now add noise:
set.seed(123)
Y <- addNoise(X, method="additive", noise=0.7)$xm
```

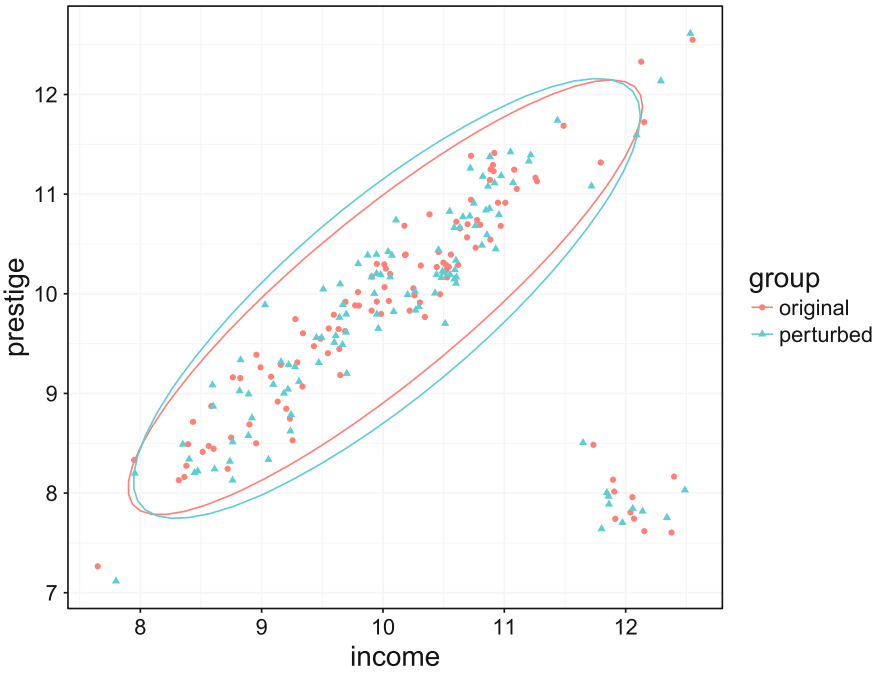


Fig. 4.4 Adding correlated noise

4.3.2.2 Correlated Additive Noise

While adding uncorrelated additive noise preserves the means and variance of the original data, co-variances and correlation coefficients are not preserved. It is preferable to apply correlated noise because the co-variance matrix of the errors is proportional to the co-variance matrix of the original data (Brand 2002).

The difference to the uncorrelated noise method is that the covariance matrix of the errors is now designed to be proportional to the covariance of the original data, i.e. $\epsilon \sim N(0, \Sigma_\epsilon = c\Sigma)$. The following holds

$$\Sigma_Z = \Sigma + c\Sigma = (1 + \alpha)\Sigma \quad . \tag{4.3}$$

Whenever the constant c is known, the covariance of the original data can be estimated from the perturbed data set.

Figure 4.4 presents the original data as well as the perturbed data set with the additive correlated noise method. Again, more noise as probably necessary is added to better see the effect of the method. Nevertheless, at least the covariance structure of this toy data set is well preserved.

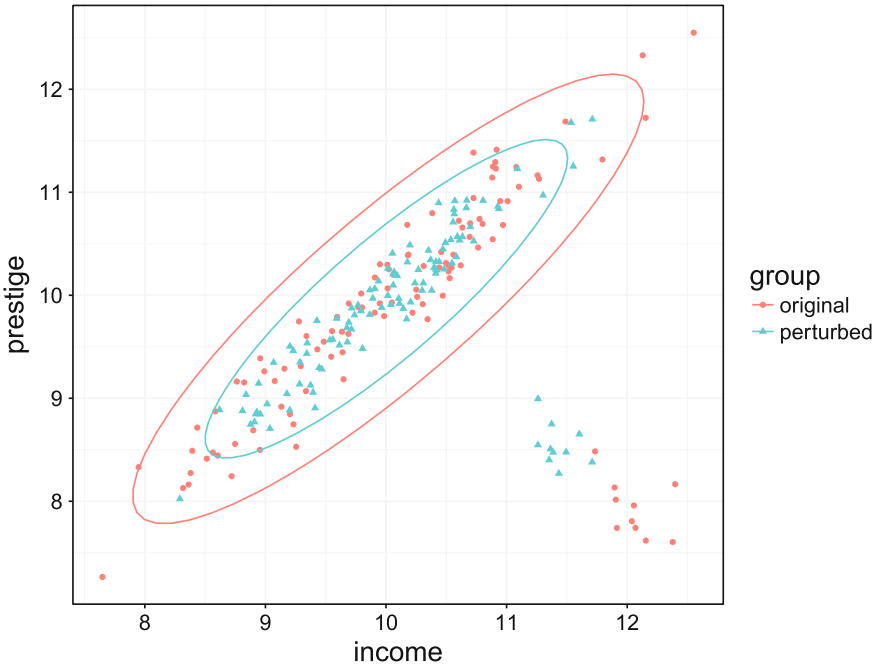


Fig. 4.5 Adding correlated noise based on transformations

4.3.2.3 Adding Correlated Noise Based on Transformations

Adding correlated noise based on transformations (Kim 1986) is another method that is also implemented in **sdcMicro**. Here we calculate $d = (1 - c^2)\epsilon$ and then $x_j d + cz_j$ whereas y_j are random numbers from $N(\frac{(1-d)\bar{x}_j}{c}, s_j)$, with $\bar{x}_j$ and s_j the mean standard deviation of x_j.

Another similar method which takes the sample size into account is described, for example, in Brand (2002). It is a method which is often denoted as the restricted correlated noise method and which is implemented in sdcMicro as well (method `restr.`).

Figure 4.5 presents the original data as well as the perturbed data set with the transformation based additive correlated noise method. Again, more noise as probably necessary is added to better see the effect of the method. The tails of the distribution are wider for the perturbed data and also a bias is introduced for the outlying group in the bottom-right of the graphic.

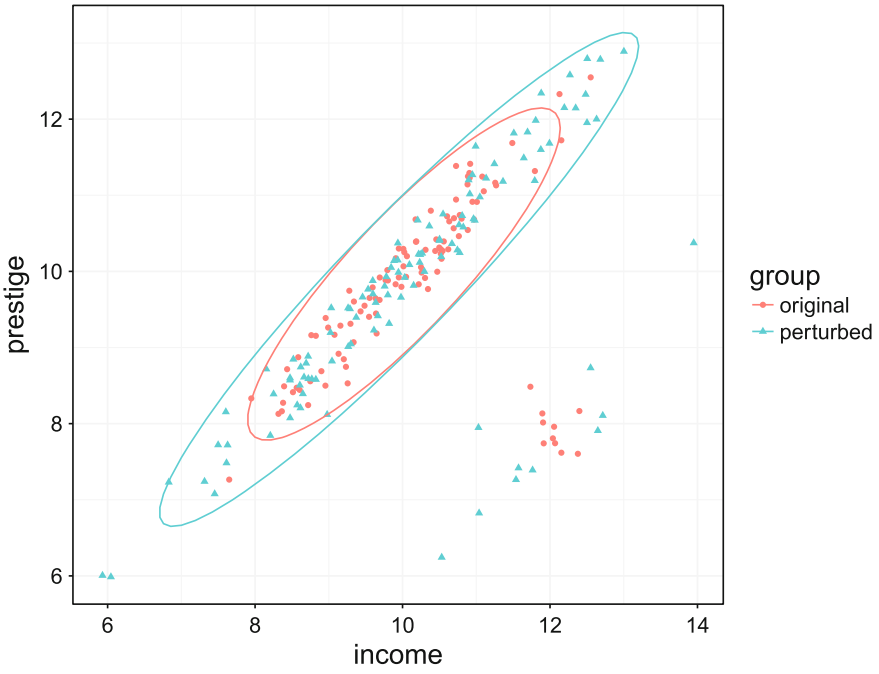


Fig. 4.6 Adding noise by Random Orthogonal Matrix Masking

4.3.2.4 Random Orthogonal Matrix Masking

Furthermore, for method ROMM (Random Orthogonal Matrix Masking) (Ting et al. 2005) perturbed data are obtained by $\mathbf{Z} = \mathbf{AX}$, whereas $\mathbf{A}$ is randomly generated and fulfils $\mathbf{A}^{-1} = \mathbf{A}^T$ (orthogonality condition). To obtain an orthogonal matrix as described in Ting et al. (2005) the Gram-Schmidt procedure (see, e.g., Golub and Van Loan 1996) is used.

Figure 4.6 presents the original data as well as the perturbed data set using ROMM. Again, more noise as probably necessary is added (parameter p) to better see the potential flaws of the method. The tails of the distribution are more narrow for the perturbed data.

4.3.2.5 Robustness Issues

The distribution of the original variables $\mathbf{x}_j$ may not follow a normal distribution and, especially the correlated noise methods may be influenced by this fact.

In this case, a robust version of the correlated noise method is described in detail by Templ and Meindl (2008). Hereby, the covariance matrices are estimated with robust covariance estimators such as the MCD estimator (Rousseeuw and Van Driessen 1999).

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.

4.3.2.6 Applying Adding Noise Methods

Of course, any mentioned noise methods can be applied to objects from class *sdcMicroObj* using function `addNoise`, e.g.

We now want to apply a method for adding correlated noise based on non-perturbed data after we undo microaggregation on our previously generated `sdc` object:

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

   ## Numerical key variables: expend, income, savings
   ##
   ## Disclosure risk is currently between [0.00
   ##
   ## Current Information Loss:
   ##   - IL1: 0.11
   ##   - Difference of Eigenvalues: 0.060
   ## ----------------------------------------------------------------------
```

We see that the data utility measure is comparable with the microaggregation on strata in the previous code chunk but the risk is higher than before using microaggregation.

Other methods ensure that the amount of noise added takes into account the underlying sample size and sampling weights.

4.3.3 Shuffling

Shuffling (Muralidhar and Sarathy 2006) generates new values for selected sensitive variables based on the conditional density of sensitive variables given non-sensitive variables. As a rough illustration, assume we have two sensitive variables, income and savings, which contain confidential information. We first use age, occupation, race and education variables as predictors in a regression model to simulate a new set of values for income and savings. We then apply reverse mapping (i.e., shuffling) to replace ranked new values with the ranked original values for income and savings. This way, the shuffled data consists of the original values of the sensitive variables. Moreover, Muralidhar and Sarathy (2006) showed that since we only need the rank of

the perturbed value in this approach, instead of generating a new value, shuffling can be implemented using only the rank order correlation matrix (measuring the strength of the association between the ranked sensitive variables and ranked non-sensitive variables) and the ranks of values of non-sensitive variables.

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()`.

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

   ## Numerical key variables: expend, income, savings
   ##
   ## Disclosure risk is currently between [0.00%; 0.22%]
   ##
   ## Current Information Loss:
   ##   - IL1: 1.63
   ##   - Difference of Eigenvalues: 2.090%
   ## ------------------------------------------------------------------------
```

To find a good model is essential to provide good results. If we would for example add variable `walls` to the model, the results are no longer of high quality.

Exercises:

Question 4.3 **Adding noise**:

Take the data set `EIA`. Assume that variables `RESSALES`, `COMSALES`, `INDSALES`, and `OTHRSALES` are continuous key variables, and `UTILNAME` and `STATE` the categorical key variables. Define your *sdcMicroObj* object. Compare additive noise and the method called `correlated2` for adding noise in terms of disclosure risk. Remark: In the next chapter, we will also compare the data utility.

Question 4.4 **Microaggregation**

Apply microaggregation on each state independently.

Question 4.5 **Microaggregation**

Assume that in general your management defines the disclosure risk on the basis of 3-anonymity for your categorical key variables. Is your data set *safe* if you apply microaggregation on your continuous key variables and ensure 3-anonymity for your categorical key variables?

References

Kowarik, A., Templ, M., Meindl, B., & Fonteneau, F. (2013). *sdcMicroGUI: Graphical user interface for package sdcMicro*. R package version 1.1.1. http://CRAN.R-project.org/package=sdcMicroGUI.

LeFevre, K., DeWitt, D. J., & Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *ICDE '06: Proceedings of the 22nd international conference on data engineering* (p. 25).

Gouweleeuw, J., Kooiman, P., Willenborg, L., & De Wolf, P.-P. (1998). Post randomisation for statistical disclosure control: Theory and implementation. *Journal of Official Statistics14*(4), 463–478.

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

Templ, M., & Meindl, B. (2008b). Robustification of microdata masking methods and the comparison with existing methods. In *Privacy in statistical databases*. Lecture Notes in Computer Science (Vol. 5262, pp. 113–126). Springer.

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

Templ, M., Meindl, B., & Kowarik, A. (2015). Statistical disclosure control for micro-data using the R package sdcMicro. *Journal of Statistical Software*, *67*(1), 1–37.

Gower, J. C. (1971a). A general coefficient of similarity and some of its properties. *Biometrics*, *27*(4), 857–871.

Hundepool, A., et al. (2007). *Handbook on statistical disclosure control*.

Brand, R. (2002). *Microdata protection through noise addition*. Lecture Notes in Computer Science London: Springer.

Kim, J. J. (1986). A method for limiting disclosure in microdata based on random noise and transformation. In *Proceedings of the section on survey research methods* (pp. 303–308). American Statistical Association.

Ting, D., Fienberg, S., & Trottini, M. (2005). Romm methodology for microdata release. In *Monographs of official statistics: Work session on statistical data confidentiality*. Luxembourg: Eurostat.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations* (3rd ed.). Baltimore: Johns Hopkins University Press.

Rousseeuw, P. J., & Van Driessen, K. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, *41*, 212–223.

Muralidhar, K., & Sarathy, R. (2006). Data shuffling—A new masking approach for numerical data. *Management Science*, *52*(2), 658–670.

Chapter 5
Data Utility and Information Loss

Abstract Once SDC methods have been applied to modify the original data set and to lower the disclosure risk, it is critical to measure the resulting information loss and data utility. Basically, two different kinds of complementary approaches exist to assess information loss: (i) direct measuring of distances/frequencies between the original data and perturbed data, and (ii) comparing statistics computed on the original and perturbed data. The first concept is common but often of limited use. The latter concept is closer to the users and data sets since its aim is to measure the differences for the most important indicators/estimates.

5.1 Element-Wise Comparisons

5.1.1 Comparing Missing Values

Missing values (NA) might be accounted for using a simple utility measure that counts the missing values in the original data $\mathbf{X}$ and the anonymized data $\mathbf{Y}$. Let $\mathbf{R}^{(X)}$ and $\mathbf{R}^{(Y)}$ be (indicator) matrices of the same size as $\mathbf{X}$ and $\mathbf{Y}$. A cell/element of $\mathbf{R}^{(X)}$ is 1 when $\mathbf{X}$ has a missing value on that position, and zero if the corresponding value is not missing. The same is with $\mathbf{R}^{(Y)}$ regarding $\mathbf{Y}$. Thus the matrices $\mathbf{R}^{(X)}$ and $\mathbf{R}^{(Y)}$ consist of zeros and ones depending on the position of missings in $\mathbf{X}$ and $\mathbf{Y}$. Let $\mathbf{R}$ a matrix of the same size with elements

$$r_{ij} = \begin{cases} 0 & \text{if } r_{ik}^{(X)} = r_{ik}^{(Y)} = 0 \quad , \\ 1 & \text{if } r_{ik}^{(X)} = 1 \ \wedge \ r_{ik}^{(Y)} = 1 \quad , \\ 0 & \text{if } r_{ik}^{(X)} = 0 \ \wedge \ r_{ik}^{(Y)} = 1 \quad , \\ 0 & \text{if } r_{ik}^{(X)} = 1 \ \wedge \ r_{ik}^{(Y)} = 0 \quad . \end{cases} \tag{5.1}$$

Hereby, it is assumed that an original data set X always has more or equal information as the anonymized data set Y. Thus, a possible imputed value in Y will not effect the following measure. We can now count the number of additional missings per variable caused by anonymizing the data using the indicator matrix $\mathbf{R}$ with n

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_5

observations and p variables,

$$m_j = \sum_i^n r_{ij} \quad , \; j \in \{1, \ldots, p\} \quad . \tag{5.2}$$

This can also be seen relatively by dividing by the number of observations, or as percentages of new missing values in each variable.

$$mp_j = 100 \cdot \frac{m_j}{n} \quad . \tag{5.3}$$

The higher m_j (or mp_j) the higher the information loss.

Let us illustrate this again on the EU-SILC data. For demonstration purposes we only define four key variables and apply local suppression on the categorical key variables.

```
library("laeken")
data("eusilc")
sdc <- createSdcObj(eusilc,
          keyVars = c("db040", "hsize", "pb220a",
                      "rb090"),
          weightVar = "rb050", hhId = "db030")
sdc <- kAnon(sdc)
print(sdc, "ls")


   ## Local suppression:
   ##  KeyVar | Suppressions (#) | Suppressions (%)
   ##   db040 |                 0 |            0.000
   ##   hsize |                 9 |            0.061
   ##  pb220a |                 0 |            0.000
   ##   rb090 |                 0 |            0.000
   ## ---------------------------------------------------------------
```

From this result it can be seen the number of missing values is the same for `hsize`, `pb220a` and `rb090`, but for key variable `db040` there are 4 additional missing values in the anonymized data set.

5.1.2 *Comparing Aggregated Information*

Instead of the direct comparison of the values of categorical variables, an alternative is to compare a contingency table $\mathbf{T}^{(\mathbf{X})}$ calculated from categorical variables of the original data $\mathbf{X}$ and the contingency table $\mathbf{T}^{(\mathbf{Y})}$ from the anonymized data $\mathbf{Y}$. More precisely, the normed sum of the absolute distances between the cells of the tables with n_1 rows and n_2 columns (see also the description from Domingo-Ferrer 2009),

$$UT = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \left| T_{ij}^{(\mathbf{X})} - T_{ij}^{(\mathbf{Y})} \right| \quad . \tag{5.4}$$

The higher UT the lower the data quality.

However, another normalization approach could be to consider only the relative change in each cell (in percentages).

$$UT2 = 100 \cdot \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \left| \frac{T_{ij}^{(\mathbf{X})} - T_{ij}^{(\mathbf{Y})}}{T_{ij}^{(\mathbf{X})}} \right| \quad . \tag{5.5}$$

In the following code, a contingency table of $rb090 \times db040$ (gender $\times$ federal state) is computed for the anonymized data and original data. First, an object of class *sdcMicroObj* is created, and then we apply PRAM on federal state.

Let's start from the beginning. First we create the **sdcMicro** object, then we apply PRAM and compare the original and table considering the prammed variable `db040` (region).

```
library("laeken")
data("eusilc")
X <- Y <- eusilc
sdc <- createSdcObj(X,
          keyVars = c("db040", "hsize", "pb220a",
                      "rb090", "pl030", "age"),
          pramVars = "db040",
          weightVar = "rb050", hhId = "db030")
sdc <- pram(sdc)
Y <- extractManipData(sdc)
```

We now compare the tables according to Eq. (5.5).

```
ct <- c("rb090", "db040")
Tx <- table(X[, ct])
Ty <- table(Y[, ct])
Tx
```

```
##         db040
## rb090    Burgenland Carinthia Lower Austria Salzburg Styria
##   male          261       517          1417      440   1128
##   female        288       561          1387      484   1167
##         db040
## rb090    Tyrol Upper Austria Vienna Vorarlberg
##   male     650          1363   1132        359
##   female   667          1442   1190        374
```

```
Ty
```

```
##         db040
## rb090    Burgenland Carinthia Lower Austria Salzburg Styria
```

```
##   male          266       514          1425       426   1116
##   female        290       559          1397       474   1177
##          db040
## rb090     Tyrol Upper Austria Vienna Vorarlberg
##   male      649          1368   1129         374
##   female    654          1434   1198         377

n1 <- nrow(Ty)
n2 <- ncol(Ty)
## UT
sum(abs(Tx - Ty)) / (n1 * n2)


## [1] 7.333333

## UT2
sum(abs(Tx - Ty)/Tx) / (n1 * n2) * 100


## [1] 1.163519
```

We see that the mean difference in the cell values of the tables is approximately 1%.

However, contingency tables are of compositional nature (Egozcue et al. 2015) since no negative cell values are possible and not any cell can be larger than the sum of cells or the corresponding marginals in the table. To consider this, one can also compare the cells using Aitchison distances, i.e. since compositional data are represented only in the simplex sample space, we have to use a different distance measure, like the Aitchison distance. It is defined for two compositions $\mathbf{x}_i = (x_1, ..., x_{n_2})$ and $\mathbf{y}_i = (y_1, \ldots, y_{n_2})$ as

$$d_a(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \left(\frac{x_i}{x_j} - \frac{y_i}{y_j} \right)^2} \quad . \tag{5.6}$$

Thus, the Aitchison distance takes care of the property that compositional data include their information only in the ratios between the parts.

We define the relative Aitchison distance as a measure of data utility

$$UTA = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} d_A \left(\mathbf{T}_i^{(\mathbf{X})}, \mathbf{T}_i^{(\mathbf{Y})} \right) \quad , \tag{5.7}$$

where $\mathbf{T}_i$ again denotes the i-th row of the contingency table obtained from the original and the anonymized data.

This is easy to be computed using the R package **robCompositions** (Templ et al. 2011).

```
library("robCompositions")
## UTA
```

```
aDist(Tx, Ty)
```

```
## [1] 0.09068296
```

UTA is theoretically sound, but the disadvantage is that the interpretation is more difficult than for *UT2*, which is computed without considering the simplex space of a contingency table.

Remark: For recoding variables, it is quite usual to join categories in categorical key variables. Doing so, it is no longer possible to compare the whole contingency tables since they are of different size as soon as less categories are present in the anonymized file.

Let's start from the beginning. First we create the **sdcMicro** object, then we do a recoding of `hsize` (household size).

```
library("laeken")
data("eusilc")
X <- eusilc
X$hsize <- factor(X$hsize)
sdc <- createSdcObj(X,
          keyVars = c("db040", "hsize", "pb220a",
                      "rb090", "pl030", "age"),
          numVars = c("age","eqIncome"),
          weightVar = "rb050", hhId = "db030")
before <- paste(6:9)
after <- "6-9"
sdc <- groupAndRename(sdc, var="hsize", before=before,
                      after=after)
Y <- extractManipData(sdc)
```

In the following the data utility measures (UT, UT2 and UTA) are calculated.

```
ct <- c("rb090", "hsize")
Tx <- table(X[, ct])
Ty <- table(Y[, ct])
n1 <- nrow(Ty)
n2 <- ncol(Ty)
## UT
sum(abs(Tx[1:n1, 1:n2] - Ty)) / (n1 * n2)
```

```
## [1] 29.83333
```

```
## UT2
sum(abs(Tx[1:n1, 1:n2] - Ty)/Tx[1:n1, 1:n2]) / (n1 * n2) * 100
```

```
## [1] 9.478475
```

```
## UTA
aDist(Tx[1:n1, 1:n2], Ty)
```

```
## [1] 0.8215698
```

The values in the cells differ about 10% and the Aitchison-based differences are much higher than in the previous example.

Contingency tables can also be visualized using mosaic plots and the multivariate dependencies of categorical data can be shown. Mosaic plots, introduced by Hartigan and Kleiner (1981), are graphical representations of multi-way contingency tables. The frequencies of the different cells of categorical variables are visualized by area-proportional rectangles (tiles). For constructing a mosaic plot, a rectangle is first split vertically at positions corresponding to the relative frequencies of the categories of a corresponding variable. Then the resulting smaller rectangles are again subdivided according to the conditional probabilities of a second variable. This can be continued for further variables accordingly. Hofmann (2003) provides an excellent description of the construction of mosaic plots and the underlying mathematical theory, and **vcd** package in R with its strucplot framework (Meyer et al. 2006) gives a well-performing implementation of mosaic plots into the hand of the users.

Figure 5.1 shows the relative frequencies for all combinations of `rb090` (sex), `pb220a` (citizenship) and `hsize` (household size). Typically, fewer larger households exist than small households as well as most respondents are from Austria for the Austrian EU-SILC data. The mosaic plots in Fig. 5.1 are produced as follows.

```
require(vcd)
ct <- c("rb090", "pb220a", "hsize")
Tx <- table(X[, ct])
Ty <- table(Y[, ct])
par(mfrow=c(1,2))
## mosaic for original data
mosaic(Tx)
## mosaic for perturbed data
mosaic(Ty)
```

However, in case of complex survey data sampled from finite populations, the population frequencies should be calculated instead of sample frequencies, i.e. Horwitz-Thompson estimates of the counts are needed. Here the function `spTable` and `spMosaic` from package **simPop** (Templ et al. 2017) could be used. The result for population frequency counts for household size $\times$ region is shown in Fig. 5.2. They are produced with the following code.

```
ct <- c("hsize", "db040")
Tx_pop <- tableWt(X[, ct], weights = X$rb050)
Ty_pop <- tableWt(Y[, ct], weights = X$rb050)
par(mfrow=c(1,2))
## mosaic for original data
mosaic(Tx_pop)
## mosaic for perturbed data
mosaic(Ty_pop)
```

In both Figures the recoding of the large households is clearly visible.

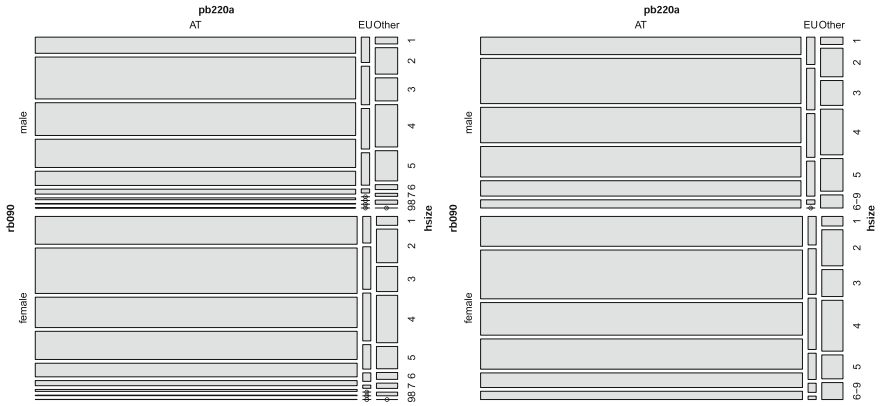


Fig. 5.1 Mosaic plot of gender (`rb090`) × citizenship (`pb220a`) × household size (`hsize`) showing the original sample frequencies (*left plot*) and the sample frequencies from the perturbed data (*right plot*)

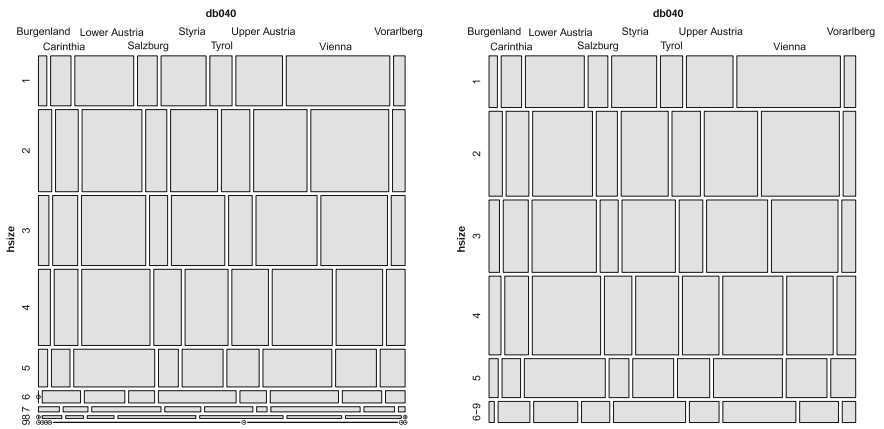


Fig. 5.2 Mosaic plot of gender (`rb090`) × citizenship (`pb220a`) × household size (`hsize`) showing the original sample frequencies (*left plot*) and the sample frequencies from the perturbed data (*right plot*)

5.2 Element-Wise Measures for Continuous Variables

In literature, element-wise measures are not often mentioned. For example, Winkler (1998) defines a microdata set to be analytically valid if (1) means and covariances on a small set of subdomains are approximately preserved, (2) marginal values for a few tabulations of the data are approximately the same and (3) at least one distributional characteristic is preserved. In addition, he proposed that a microdata file is analytically interesting if six variables on important subdomains can be validly analyzed. Also Hundepool et al. (2012) suggest to compare means, covariances, correlations,

loadings from a principal component analysis, etc. In **sdcMicro**, a measure called **eigen** is estimated, calculating relative absolute differences between eigenvalues of the co-variances 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. However, all these suggestions and proposed measures are quite general, and may be not suitable for every data set. For example, covariance-based measures are only suitable in the multivariate context without any missing values and zeros in the data.

After discussing element-wise measures for continuous variables, this book focuses on specific utility measures in Sect. 5.5 that are focusing on quality of indicators that are data-dependent and subject matter dependent. General methods as like covariance-based methods are not mentioned because of limited use.

In any case, element-wise comparisons of values from the original and the anonyized data requires the definition of a distance. Element-wise measures of information loss and data utility can be, for example, be based on the classical or robust distances between original and perturbed values. Classical distances are covered in the definition of the generalized Gower distance in Eq. 5.11. Multivariate (robust) distances can also be considered by calculating (robust) Mahalanobis distances for continuous scaled variables of the original and the anonymized data, defined for an observation $\mathbf{x}_i$ as

$$MD(\mathbf{x}_i) = \left[(\mathbf{x}_i - U)^t C^{-1} (\mathbf{x}_i - U)\right]^{1/2} \quad , i = 1, \ldots, n \quad , \tag{5.8}$$

where U and C are estimators of location and covariance. Clearly, for reliable consideration if the main bulk of the data is well preserved in the sense of robust statistics, both T and C have to be estimated in a robust way, and not in the traditional way by arithmetic mean vector and sample covariance matrix. Robust estimates of location and covariance can be obtained for instance from the MCD (Minimum Covariance Determinant) estimator (Rousseeuw and Driessen 1998).

A utility measure be based on (robust) covariance matrices can be defined as follows

```
X <- eusilc[, c("age", "eqIncome")]
## add noise to age and eqIncome
sdc <- addNoise(sdc, method = "correlated2")
Y <- extractManipData(sdc)[, c("age","eqIncome")]
Y$age <- round(as.numeric(as.character(Y$age)))
require(robustbase)
covX <- covMcd(X)
covY <- covMcd(Y)
mdX <- sqrt(mahalanobis(X,
                    center = covX$center,
                    cov = covX$cov))
mdY <- sqrt(mahalanobis(Y,
                    center = covY$center,
                    cov = covY$cov))
cat("\n relative difference in percentages:",
    round(1 / length(mdX) * sum(abs(mdX - mdY) / mdX, na.rm = TRUE) *
    100, 4), "% \n")

   ##
   ##  relative difference in percentages: 11.3035 %
```

In average the Mahalanobis distances differs around 11% after recoding of age.

Following are three common proposals to measure the information loss and data utility:

- IL1s, proposed by Yancey et al. (2002), can be interpreted as the scaled distances between original and perturbed values. Again let $\mathbf{X} = \{x_{ij}\}$ be the original data set, $\mathbf{Y} = \{y_{ij}\}$ is a perturbed version of $\mathbf{X}$, and x_{ij} is the j-th variable in the i-th original observation. Both data sets consist of n observations and p variables each. The measure of information loss is defined by

$$IL1 = \frac{1}{pn} \sum_{j=1}^{p} \sum_{i=1}^{n} \frac{|x_{ij} - y_{ij}|}{\sqrt{2} S_j} \quad , \tag{5.9}$$

  where S_j is the standard deviation of the j-th variable in the original data set.
- **prediction quality** measures the differences between estimates obtained from fitting a pre-specified regression model on the original data and the perturbed data:

$$|(\bar{\hat{y}}_w^o - \bar{\hat{y}}_w^m)/\bar{\hat{y}}_w^o| \quad , \tag{5.10}$$

  with $\bar{\hat{y}}_w$ being fitted values from a pre-specified model obtained from the original (index o) and the modified data (index m). Index w indicates that the survey weights should be considered when fitting the model.

Note that two of 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 extracted. We define our *sdcMicroObj* first and then already apply a method to anonymize the continuous key variables (Fig. 5.3).

```
df <- data.frame(mdX = mdX, mdY = mdY)
require(ggplot2)
gg <- ggplot(df, aes(x = mdX, y = mdY)) + geom_point()
gg <- gg + xlab("Mahalanobis distances - original data") +
           ylab("Mahalanobis distances - perturbed data") print(gg)
```

```
library("laeken")
data("eusilc")
sdc <- createSdcObj(eusilc,
          keyVars = c("db040", "hsize", "pb220a",
                      "rb090", "pl030", "age"),
          numVars = "eqIncome",
          weightVar = "rb050",
          hhId = "db030")
sdc <- microaggregation(sdc)
```

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

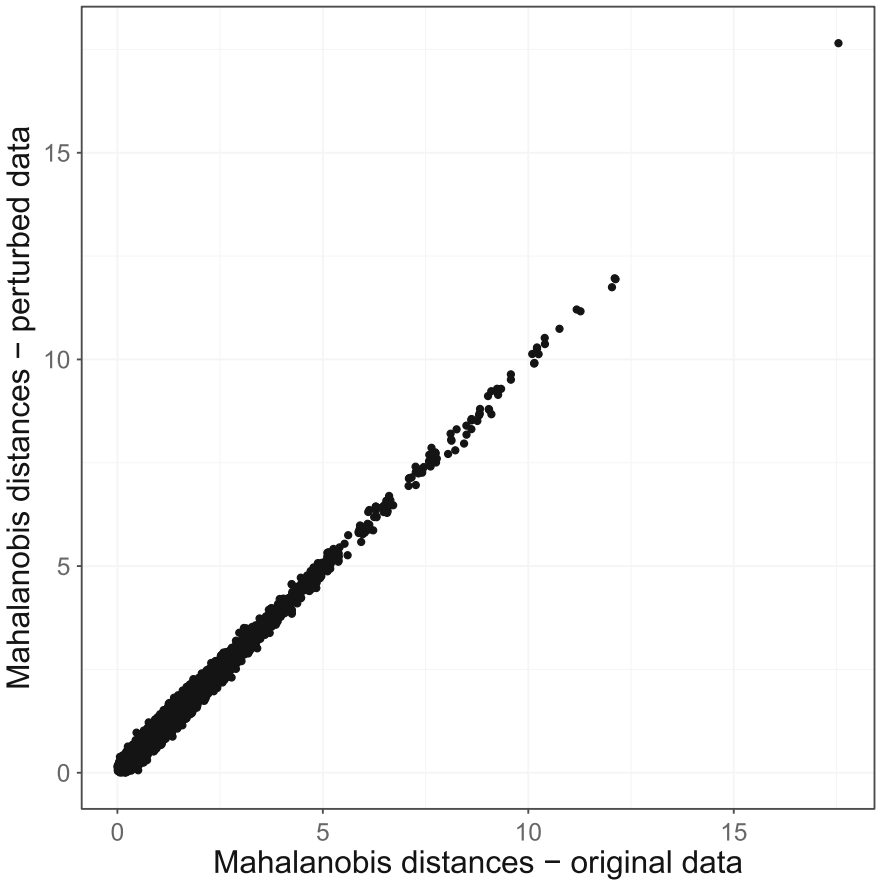


Fig. 5.3 Mahalanobis distances from original data (EU-SILC) versus the perturbed data after adding correlated noise to age and income

```
## $il1
## [1] 0.001442564
##
## $eigen
## [1] 0
```

Let's have a look at another method.

```
sdc <- undolast(sdc)
sdc <- addNoise(sdc, method = "correlated2")
get.sdcMicroObj(sdc, "utility")
```

```
## $il1
## [1] 0.1138542
##
```

```
## $eigen
## [1] 2.220446e-16
```

Since the higher the values of these measures, the lower is the data utility, we see that the data utility is lower for adding correlated noise than for microaggregation (method *mdav*). In the following code listing we see that additive noise gives much worse results.

```
sdc <- undolast(sdc)
sdc <- addNoise(sdc, method = "additive")
get.sdcMicroObj(sdc, "utility")
```

```
## $il1
## [1] 1.687381
##
## $eigen
## [1] 2.220446e-16
```

5.2.1 Element-Wise Comparisons of Mixed Scaled Variables

A distance measure that takes different scales of variables into account is based on an extension of the Gower distance (Gower 1971), which can handle distance variables of the type binary, categorical, ordered, continuous and semi-continuous. The distance between two observations is the weighted mean of the contributions of each variable, where the weight should represent the importance of the variable.

The observations of a data matrix $\mathbf{X}$ (the original data) and the observations of the anonymized data $\mathbf{Y}$ are collected in the rows $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^t$ and $\mathbf{y}_i = (y_{i1}, \ldots, x_{ip})^t$ respectively (for $i = 1, \ldots, n$ and p the number of variables).

The distance between the i-th observation $\mathbf{x}_i$ and $\mathbf{y}_i$ can be defined as

$$d(\mathbf{x}_i, \mathbf{y}_i) = \frac{\sum_{k=1}^{p} w_k \delta_{i,k}}{\sum_{k=1}^{p} w_k}, \tag{5.11}$$

where w_k is the weight and $\delta_{i,k}$ is the contribution of the k-th variable.

For continuous variables the absolute distance divided by the total range is used

$$\delta_{i,k} = |x_{i,k} - y_{i,k}|/r_k, \tag{5.12}$$

where $x_{i,k}$ and $y_{i,k}$ are the values of the k-th variable in the original and anonymized data of the i-th observation. r_k is the range of the k-variable in the original data set. Ordinal variables are converted to integer variables and then the absolute distance divided by the range is computed. The categories are therefore treated as if they were equidistant.

For nominal and binary variables a simple 0/1 distance is used

$$\delta_{i,k} = \begin{cases} 0 & \text{if } x_{i,k} = y_{i,k} \\ 1 & \text{if } x_{i,k} \neq y_{i,k} \end{cases}, \quad . \tag{5.13}$$

Another special type of variables are semi-continuous variables, consisting of continuously distributed part and probability mass at one point. An example for such a variable might be an income component, which is 0 for some observations and continuously distributed in the remaining observations. The contributions for semi-continuous variables is computed as mixture of the contribution for nominal and continuous variables

$$\delta_{i,k} = \begin{cases} 0 & \text{if } x_{i,k} = s_k \wedge y_{i,k} = s_k \\ 1 & \text{if } x_{i,k} \neq s_k \wedge y_{i,k} = s_k \\ 1 & \text{if } x_{i,k} = s_k \wedge y_{i,k} \neq s_k \\ |x_{i,k} - y_{i,k}| / r_k^{(\mathbf{x}_k)} & \text{if } x_{i,k} \neq s_k \wedge y_{i,k} \neq s_k \end{cases}, \tag{5.14}$$

where s_k is the special value for the k-th variable, e.g., the 0 in the income variable example.

As described above, all $\delta_{i,k}$ are in $[0, 1]$, as a consequence the computed distances $d(\mathbf{x}_i, \mathbf{y}_i)$ between two observations are also inside this interval.

In general, the higher the distance in Eq. (5.11), the lower the data utility.

```
library("laeken")
library("VIM")
data("eusilc")
X<-Y <- eusilc
sdc<- createSdcObj(X,
          keyVars = c("db040", "hsize", "pb220a",
                      "rb090", "pl030", "age"),
          numVars = c("eqIncome"),
          pramVars = "db040",
          weightVar = "rb050", hhId = "db030")
sdc <- pram(sdc)
sdc <- microaggregation(sdc, strata="db040")
Y <- extractManipData(sdc)
gd <- VIM:::gowerD(X[, c("eqIncome","db040")], Y[, c("eqIncome","db040")],
              numerical="eqIncome", factors="db040",
              orders = NULL, mixed=NULL, levOrders = NULL,
              weights=rep(1,2), mixed.constant = 0)
sum(abs(gd)) / nrow(eusilc) * 100
```

5.3 Entropy

The alternative option is to use an entropy function. Given $c_1, c_2, \ldots, c_k$ categories of a variable $\mathbf{X_j}$, the entropy E_{c_j} is defined as

$$E_{c_j} = -\frac{1}{n} \sum_{c_j \in \mathbf{X_j}} f_{c_j} log\left(\frac{f_{c_j}}{n}\right) \quad , \tag{5.15}$$

where f_{c_j} is the frequency of category c_j of variable $\mathbf{X}_j$ and n the total number of observations.

This could also be used to suppress variables, e.g. the variable with the lowest value of the entropy function is chosen for further global recoding (*age* in our case, see below).

```
## entropy of key variables on original data X
entropy <- function(fk, n){
  1 / n  * sum(fk * log(fk / n))
}
## for hsize
n <- nrow(eusilc)
fk <- as.numeric(table(eusilc$hsize))
entropy(fk, n)
```

```
## [1] -1.765339
```

```
## for age
entropy(as.numeric(table(eusilc$age)), n)
```

```
## [1] -4.440551
```

```
## for pb220a
entropy(as.numeric(table(eusilc$pb220a)), n)
```

```
## [1] -0.4446661
```

5.4 Propensity Score Methods

Woo et al. (2009) propose the use of a method that is based on the idea of propensity scores. Propensity scores are usually used in medical studies where the propensity score is the probability of being assigned to treatment given covariate variables. From this theory, it can be said that treatment assignment and covariates are conditionally independent given the propensity scores. The groups should have similar distributions of covariates, if they have the same distributions of propensity scores.

The idea is to merge/join the original and the perturbed data sets and then create a new index variable with ones for the original data and zeros for observations from anonymized data. A logistic regression model is then fitted using the new index variable as the response variable. Predictions from this model are then compared with the proportion of observations of the perturbed data to the original data (usually 1/2).

Woo also describes two other measures, one based on cluster analysis (evaluating the cluster sizes) and another which compares the empirical cumulative distribution function. They concentrate only on data utility measures and do not account for disclosure risk. Karr et al. (2006) proposes measures based on differences between inferences on original and perturbed data that are tailored to normally distributed data and they used also the propensity score method in Oganian and Karr (2006).

Let us have a closer look at (Woo et al. 2009) with a practical example. Again, the `eusilc` data set used is

```
eusilc$hsize <- factor(eusilc$hsize)
## create sdcMicroObj
sdc <- createSdcObj(eusilc,
          keyVars = c("db040", "hsize", "pb220a",
                      "rb090", "pl030", "age"),
          pramVars = "db040",
          numVars = "eqIncome",
          weightVar = "rb050", hhId = "db030")
## apply first anonymization
sdc <- pram(sdc)
Y <- extractManipData(sdc)
## apply second and third anonymization
before <- paste(6:9)
after <- c("6-9")
sdc <- groupAndRename(sdc, var="hsize", before=before,
                  after=after)
sdc <- microaggregation(sdc)
Y2 <- extractManipData(sdc)
```

The original and the perturbed data are joined together (row binding).

```
## to benchmark, original and original
Z <- rbind(eusilc, eusilc)
## original and perturbed version 1
ZA <- rbind(eusilc, Y)
## original and perturbed version 2
ZA2 <- rbind(eusilc, Y2)
```

Next the index vector indicating is created, determining if an observation belongs to the original data or the perturbed data.

```
Z$index <- ZA$index <- ZA2$index  <- rep(0:1, each=nrow(eusilc))
```

We then fit the following model. For simplicity we choose a very simply model. However, we recommend to improve the model by carefully searching for the best model.

```
form <- as.formula("index ~ db040 + hsize + pb220a +
                    rb090 + pl030 + eqIncome")
res <- glm(form, data=Z, family = binomial())
resA <- glm(form, data=ZA, family = binomial())
resA2 <- glm(form, data=ZA2, family = binomial())
```

First we can look on how many one's and zero's are predicted. If the model fits perfect, the ratio between one's and zero's should be one.

```
ps <- function(mod){
  p <- predict(mod, type="response")
  t1 <- as.numeric(table(p < 0.5))
  return(t1[1] / t1[2])
}
p <- ps(res)
p
```

```
## [1] 0.8302343
```

```
pa <- ps(resA)
pa
```

```
## [1] 1.19668
```

```
pa2 <- ps(resA2)
pa2
```

```
## [1] 1.185971
```

Ideally `ps(res)` should be 1 or very close to one. In this example we see that the more variables are modified, the higher the ratio between (0/1) in the predictions. Note that the results may get better when selecting the best possible model by possible also considering also interaction terms.

Woo et al. (2009) compares also the distribution of the predicted values by

$$UP = \frac{1}{n_m} \sum_{i=1}^{n_m} (p_i - c)^2 \quad , \tag{5.16}$$

where n_m is the number of observations in the merged data set, p_i is the estimated probabilities being in group 1 (original data) or group 2 (perturbed data). c is usually determined as 0.5 whenever the perturbed data have the same amount of observations as the original data. If UP is close to zero, the data utility is high. The worse case is if $UP \sim 1/4$, where the two data sets are completely distinguishable (see also Woo et al. 2009).

```
1 / nrow(Z) * sum((predict(res, type="response") - 0.5)^2)
```

```
## [1] 9.055325e-30
```

```
1 / nrow(Z) * sum((predict(resA, type="response") - 0.5)^2)
```

```
## [1] 1.097417e-05
```

```
1 / nrow(Z) * sum((predict(resA2, type="response") - 0.5)^2)
```

```
## [1] 0.01095378
```

We see that the grouping of the large households and the microaggregation of the equivalized income has a large effect on *UP*, while only to apply PRAM on variable `db040` (federal states) has no big effect and the data utility is reported to be high.

Excercises:

Question 5.1 As done already before, produce an object of class *sdcMicroObj* using the data set `eusilc`. Apply some recodings and local suppression for categorical key variables, and apply a method to anonymize the continuous key variable `eqIncome`. Cross tabulate `hsize`, `rb090` (gender) and `pb220a` (citizenship) and compare this table with the result from the raw survey data.

Question 5.2 Plot the equivalized income from the raw survey data against the modified equivalized income. Split the plot by conditioning on `pl030`.

5.5 Quality Indicators

Although, in practice, it is not possible to create a file with the exact same structure as the original file after applying SDC methods, an important goal of SDC should be to minimize the difference in the statistical properties of the perturbed data and the original data. It is a fact that not every estimate can be preserved. Therefore the aim is to preserve the most important estimates, i.e. instead of calculating generally defined utility measures as in the previous sections, evaluation on context/data dependent indicators is proposed. Such an approach to measuring data utility is based on benchmarking indicators (Ichim and Franconi 2010; Templ 2011a, 2015).

5.5.1 General Procedure

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 microdata 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 main indicators calculated from the protected data differ significantly from those estimated from the original data set, the SDC procedure should be restarted. It is possible to either change some parameters of the applied methods or start from scratch and completely change the choice of SDC methods. The benchmarking indicator approach is usually applied to assess the impact of SDC methods on continuous variables. But it is also applicable to categorical variables. In addition, the benchmarking indicators approach can be applied to subsets of the data. In this case, benchmarking indicators are evaluated for each of the subsets and the results are evaluated by reviewing differences between indicators for original and modified data within each subset.

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.

5.5.2 *Differences in Point Estimates*

For an object of class *sdcMicroObj*, 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.

For the EU-SILC data, one of the most important indicators is the Gini coefficient. Therefore, this indicator is used to evaluate the quality of the anonymized data set. Given a vector $x_1, \ldots, x_n$ with sample weights $w_1, \ldots, w_n$ the Gini coefficient can be estimated by

$$\widehat{Gini} = \frac{2\sum_{i=1}^{n}\left(w_i x_i \sum_{j=1}^{i} w_i\right) - \sum_{i=1}^{n} w_i^2 x_i}{\left(\sum_{i=1}^{n} w_i\right)\sum_{i=1}^{n} w_i x_i} - 1 \quad .$$

The Gini coefficient takes on values between 0 and 1. A value of 0 stand for perfect equality, meaning that every data point in the sample has the same value or every

individual has the same income or volume of a certain good. With a value of 1 the Gini coefficient would indicate perfect inequality, meaning that all but one data point are equal to zero or that one individual has all the income or volume of certain good.

The following line of code estimates the Gini coefficient and stores the result within the `sdcMicroObj` `sdc`.

```
sdc@additionalResults$gini <- gini(inc = "eqIncome",
       weigths = "rb050",
       breakdown = "db040",
       data = extractManipData(sdc))$valueByStratum$value
```

This result can then be simple compared for each category of `db040` (federal state) with the result of the original data to get an indicator of data utility expressed as relative absolute errors (in percentages):

```
res <- gini(inc = "eqIncome",
            weigths = "rb050",
            breakdown = "db040",
            data = eusilc)$valueByStratum$value
100*abs((res - sdc@additionalResults$gini)/res)


   ## [1] 2.173689 2.386133 0.270414 1.289537 1.055491 1.080695
   ## [7] 1.925351 1.468171 4.996778
```

It can be seen that in few countries the change in the Gini coefficient is about 3% of its absolute value, in other regions, the differences are smaller.

5.5.3 Differences in Variances and MSE

However, it is also recommended to not look only on point estimates but also estimate the variances and compare the variances obtained from the original and the perturbed data.

The estimation of variance is not always an easy task when working with complex survey collected based on complex sample designs. A calibrated bootstrap (Alfons and Templ 2013) to estimate the variances is applied in the following.

```
res <- gini(inc = "eqIncome",
            weigths = "rb050",
            breakdown = "db040",
            data = eusilc)
resVar <- variance("eqIncome", weights = "rb050",
            design = "db040", breakdown = "db040",
            data = eusilc, indicator = res, R = 50,
            X = calibVars(eusilc$db040), seed = 123)
res <- resVar$valueByStratum$value
resVar <- resVar$varByStratum$var
eusilcA <- extractManipData(sdc)
resA <- gini(inc = "eqIncome",
```

```
            weigths = "rb050",
            breakdown = "db040",
            data = eusilcA)
resVarA <- variance("eqIncome", weights = "rb050",
             design = "db040", breakdown = "db040",
             data = eusilcA, indicator = resA, R = 50,
             X = calibVars(eusilc$db040), seed = 123)
resA <- resVarA$valueByStratum$value
resVarA <- resVarA$varByStratum$var
100*abs((res - resA) / res)


   ## [1] 2.173689 2.386133 0.270414 1.289537 1.055491 1.080695
   ## [7] 1.925351 1.468171 4.996778

100*abs((resVar - resVarA) / resVar)


   ## [1] 35.123843 13.152475  1.601867 26.820150 45.971562
   ## [6] 11.283797 28.450694 18.381783 31.100778
```

A mean squared error (MSE) kind of measure would then be the differences in the point estimates (bias) to the square plus the differences in variances.

```
(res - resA)^2 + (resVar -resVarA)


   ## [1]  1.164910938  0.315197915  0.007902761  0.226787385
   ## [5] -0.013603540  0.131555209  0.3100s45368  0.218618271
   ## [9]  1.867190441
```

5.5.4 *Overlap in Confidence Intervals*

However, also the overlap of confidence intervals is of interest.

Let $[l_{c_j}^{(\mathbf{X})}, u_{c_j}^{(\mathbf{X})}]$ the 95%-confidence interval for a given parameter in the original data $\mathbf{X}$ in strata c_j and let $[l_{c_j}^{(\mathbf{Y})}, u_{c_j}^{(\mathbf{Y})}]$ the corresponding interval in the anonymized data. The intersection of the two intervals is denoted by

$$[li_{c_j}, ui_{c_j}] = [l_{c_j}^{(\mathbf{X})}, u_{c_j}^{(\mathbf{X})}] \cap [l_{c_j}^{(\mathbf{Y})}, l_{c_j}^{(\mathbf{Y})}] \tag{5.17}$$

The utility measure is then given by

$$UC = \frac{1}{2K} \sum_{j=1}^{K} \left(\frac{ui_{c_j} - li_{c_j}}{u_{c_j}^{(\mathbf{X})} - l_{c_j}^{(\mathbf{X})}} + \frac{ui_{c_j} - li_{c_j}}{u_{c_j}^{(\mathbf{Y})} - l_{c_j}^{(\mathbf{Y})}} \right) \quad , \tag{5.18}$$

Table 5.1 Lower (l) and upper (u) limits of the confidence intervals for the GDP for each category of *education* for the original data, for recoded and local suppressed data, for recoded, local suppressed and noise addition to data, for prammed and microaggregated data and for recoded, local suppressed and shuffled data

Data	ISCED 0-1	ISCED 2	ISCED 3-4	ISCED 5A	ISCED 5B
original (l)	0.15938	0.12102	0.22572	0.29568	0.21744
original (u)	0.26525	0.15023	0.23944	0.35010	0.25835
rec+ls+ma (l)	0.16123	0.12144	0.22624	0.28891	0.21290
rec+ls+ma (u)	0.27062	0.15211	0.23970	0.34381	0.25904
rec+ls+noise (l)	0.17012	0.12106	0.22399	0.29135	0.21152
rec+ls+noise (u)	0.27011	0.15172	0.23776	0.34551	0.25805
pram+ma (l)	0.17682	0.12200	0.22554	0.29064	0.21946
pram+ma (u)	0.27230	0.15065	0.24197	0.33822	0.26172
rec+ls+shuffle (l)	−0.01865	0.09365	0.18510	0.19294	0.19859
rec+ls+shuffle (u)	0.24584	0.12496	0.20950	0.25071	0.26183

Table 5.2 Coverage rates for confidence intervals of the gender pay gap in each educational sector between the original and perturbed data

Data	ISCED 0 and 1	ISCED 2	ISCED 3 and 4	ISCED 5A	ISCED 5B
rec+ls+ma	98.25	98.55	96.21	88.45	88.65
rec+ls+noise	89.85	99.86	87.81	91.58	99.26
pram+ma	83.52	96.63	83.45	78.18	95.08
rec+ls+shuffle	81.67	13.51	0.00	0.00	64.67

with K the number of strata in which the confidence intervals are being estimated. When the intervals are identical in both $\mathbf{X}$ and $\mathbf{X}$, $UC = 1$. In the other extreme case, when the intervals do not overlap at all, $UC = 0$.

As an example, the upper and lower confidence intervals for the GDP in domain *education* are given in Table 5.1. It is easy to see that the length of the confidence intervals are shorter for category ISCED 3-4 and largest for ISCED 0-1.

Again, the shuffling method does not seem to be able to give approximately the same confidence intervals.

A clearer picture is supported by Table 5.2 where the overlap of the confidence intervals for the GDP—estimated from the perturbed and the original data—is reported.

The coverage rates are relatively high for all methods except recoding+local suppression+shuffling. Differences in some categories are visible when comparing the other methods whereas no clear ranking of them in terms of quality can be made.

The coverage rates for the gender pay gap in domain *age* (Table 5.3) are similar. Mostly the recoding+local suppression+microaggregation methods performs slightly better than recoding+local suppression+adding noise and pram+microaggregation.

Table 5.3 Coverage rates for confidence intervals of the GDP in each age class between the original and perturbed data

Data	(0,19]	(19,29]	(29,39]	(39,49]	(49,59]	(59,120]
rec+ls+ma	98.81	76.40	99.28	82.41	95.82	91.45
rec+ls+noise	94.90	80.27	94.31	89.60	89.70	96.76
pram+ma	84.26	88.92	95.02	88.55	92.58	86.94
rec+ls+shuffle	0.00	32.75	0.00	0.00	0.00	0.00

Table 5.4 Coverage rates for confidence intervals of the Gini indices in each age × gender domain between the original and perturbed data

Data	(0,19]:f	(0,19]:m	(19,29]:f	(19,29]:m	(29,39]:f	(29,39]:m
rec+ls+ma	93.64	81.66	96.83	94.93	89.24	95.63
rec+ls+noise	52.71	0.00	22.12	37.18	63.09	87.92
pram+ma	88.29	82.49	88.39	93.05	85.36	94.50
rec+ls+shuffle	82.61	0.00	0.00	0.00	20.38	0.00
	(39,49]:f	(39,49]:m	(49,59]:f	(49,59]:m	(59,120]:f	(59,120]:m
rec+ls+ma	84.69	75.33	99.21	94.59	95.40	92.22
rec+ls+noise	88.49	83.07	80.52	89.70	93.94	96.03
pram+ma	97.89	85.00	96.78	82.25	88.31	94.94
rec+ls+shuffle	12.55	55.93	0.00	0.00	0.00	0.00

However, a completely different picture is seen for the absolute relative bias of the Gini index in Table 5.4. Recoding+local suppression+microaggregation outperforms all other methods. PRAM+microaggregation also gives acceptable results but recoding+local suppression+adding noise gives low coverage rates for age classes below 29 years. Shuffling results in the highest biased estimates.

5.5.5 Differences in Model Estimates

As already mentioned, to compare the regression coefficients of original and anonymized data sets, the same categories in the explanatory variables of the model must be present. Thus the recoded 12 categories of *economic activity* are used also for the original data set, keeping in mind that this means a certain kind of information loss.

In Table 5.5 the regression coefficients for the original and the anonymized data sets are shown.

Recoding+local suppression+microaggregation again performs best and the confidence intervals obtained from the anonymized data cover the confidence intervals obtained from the original data almost always completely. Almost as good is the quality of data anonymized by recoding+local suppression+adding correlated noise.

Table 5.5 Regression coefficients

	original	rec+ls+ma	rec+ls+noise	pram+ma	rec+ls+shuffle
(Intercept)	1.50454	1.52627	1.51374	1.40474	1.63726
Sexmale	0.20478	0.20484	0.20433	0.20970	0.19733
age(19,29]	0.57210	0.57190	0.58560	0.57659	0.76536
age(29,39]	0.73750	0.73745	0.75186	0.74388	0.91469
age(39,49]	0.81758	0.81746	0.83260	0.82634	0.96199
age(49,59]	0.85660	0.85597	0.87072	0.86754	0.89338
age(59,120]	0.81553	0.81067	0.82604	0.82169	0.49264
educationISCED 2	0.03692	0.02006	0.01011	0.03834	−0.25102
educationISCED 3 and 4	0.28314	0.26646	0.25737	0.28667	−0.16874
educationISCED 5A	0.73406	0.71508	0.70647	0.74198	0.09813
educationISCED 5B	0.44484	0.42802	0.41959	0.45337	−0.01251
LocationAT2	−0.07516	−0.07523	−0.07528	−0.06368	−0.00673
LocationAT3	−0.01230	−0.01207	−0.01132	−0.00900	−0.00098
NACE1D-Manufactoring	−0.05542	−0.06029	−0.05441	0.01740	−0.01600
NACE1E-Electricity	0.09709	0.09018	0.09264	0.12244	−0.02588
NACE1F-Construction	−0.12280	−0.12891	−0.12260	−0.03775	-0.01806
NACE1G-Trade	−0.18916	−0.19422	−0.18848	−0.09872	−0.02576
NACE1H-Hotels	−0.37478	−0.37962	−0.37589	−0.24398	−0.02269
NACE1I-Transport	−0.17130	−0.17632	−0.17061	−0.07943	−0.00939
NACE1J-FinancInt	0.14921	0.14532	0.15055	0.19273	−0.01993
NACE1K-RealEstate	−0.13433	−0.13901	−0.13517	−0.05156	−0.02072
NACE1M-Education	−0.16289	−0.16650	−0.16300	−0.07845	−0.02505
NACE1N-Health	−0.11299	−0.11734	−0.11360	−0.02939	−0.01838
NACE1O-Other	−0.19113	−0.19585	−0.19353	−0.10283	−0.01054

The results from invariant pram+microaggregation are good for all coefficients except those related to *economic activity*. This is not surprising since this variable was one of the variables which was changed using PRAM. Some few coefficients are well preserved from the recoding+local suppression+shuffling anonymized data, but others are not. The reason is that even if the distribution of the continuous shuffled

variables are well preserved, the relation to other variables that are not included in the shuffling model might be not preserved. A better model would probably lead to better results.

Excercises:

Question 5.3 Undo-the last step of the anonymization procedure above, apply a different disclosure limitation technique and re-calculate the Gini coefficients. Compare it with previous results.

References

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.

Domingo-Ferrer, J. (2009). Information loss measures. In L. Liu & M. Tamerzsu (Eds.), *Encyclopedia of database systems* (pp. 1499–1501). Springer US. ISBN 978-0-387-35544-3.

Egozcue, J. J., Pawlovsky, V., Templ, M., & Hron, K. (2015). Independence in contingency tables using simplical geometry. *Communications in Statistics*, *44*(18), 3978–3996.

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, *27*, 857–871.

Hartigan, J. A., & Kleiner, B. (1981). Mosaics for contingency tables. In W. F. Eddy (Ed.), *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface* (pp. 268–273). New York: Springer.

Hofmann, H. (2003). Constructing and reading mosaicplots. *Computational Statistics & Data Analysis*, *43*(4), 565–580.

Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E. S., Spicer, K., et al. (2012). *Statistical disclosure control*. Wiley Series in Survey Methodology. Wiley. ISBN 9781118348222. https://books.google.at/books?id=BGa3zKkFm9oC.

Ichim, D., & Franconi, L. (2010). Strategies to achieve SDC harmonisation at european level: Multiple countries, multiple files, multiple surveys. In *Privacy in statistical databases* (pp. 284–296).

Karr, A. F., Kohnen, C. N., Oganian, A., Reiter, J. P., & Sanil, A. P. (2006). A framework for evaluating the utility of data altered to protect confidentiality. *The American Statistician*, *60*(3), 224–232.

Meyer, D., Zeileis, A., & Hornik, K. (2006). The strucplot framework: Visualizing mulit-way contingency tables with vcd. *Journal of Statistical Software*, *17*(3), 1–48.

Oganian, A., & Karr, A. F. (2006). Combinations of SDC methods for microdata protection. In J. Domingo-Ferrer & L. Franconi (Eds.), *Privacy in statistical databases* (Vol. 4302, pp. 102–113)., Lecture Notes in Computer Science Heidelberg: Springer.

Rousseeuw, P. J., & Van Driessen, K. (1998). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, *41*, 212–223.

Templ, M. (2011a). 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.

Templ, M. (2011b). Comparison of perturbation methods based on pre-defined quality indicators. In *Joint UNECE/Eurostat work session on statistical data confidentiality*. Tarragona, Spain. Invited paper.

Templ, M. (2015, accepted for publication). Quality indicators for statistical disclosure methods: A case study on the structural earnings survey. *Journal of Official Statistics*.

Templ, M., Hron, K., Filzmoser, P. (2011) *robCompositions: An R-package for robust statistical analysis of compositional data* (pp. 341–355). Wiley. ISBN 9781119976462. http://dx.doi.org/10.1002/9781119976462.ch25.

Templ, M., Meindl, B., Kowarik, A., & Dupriez, O. (2017, accepted for publication in December 2015). Simulation of synthetic complex data: The R-package simPop. *Journal of Statistical Software*, 1–38.

Winkler, W. E. (1998). Re-identification methods for evaluating the confidentiality of analytically valid microdata. *Research in Official Statistics*, *1*, 50–69.

Woo, M., Reiter, J. P., Oganian, A., & Karr, A. F. (2009). Global measures of data utility for microdata masked for disclosure limitation. *Journal of Privacy and Confidentiality*, *1*(1), 111–124.

Yancey, W. E., Winkler, W. E., & Creecy, R. H. (2002). Disclosure risk assessment in perturbative microdata protection. In *Inference control in statistical databases*. Lecture Notes in Computer Science (pp. 49–60). Springer.

Chapter 6
Synthetic Data

Abstract The generation of synthetic data sets serves as a statistical disclosure control solution to generate public use files out of confidential/protected data. In addition, it is also a tool to create "augmented data sets" which serve as input for micro-simulation models or as data sets for remote execution. Multiple approaches and tools have been developed to generate synthetic data. These approaches can be categorized into three main groups: synthetic reconstruction, combinatorial optimization, and model-based generation. In this chapter, the most promising and important method—model-based simulation—is described in detail. It is also the reason why whole populations are simulated rather than only surveys. For other approaches, we refer to Drechsler (2011) (Drechsler, Synthetic data sets for statistical disclosure control. Springer, New York, 2011) and other references below.

6.1 Introduction

Creating a synthetic data set first involves to extract relevant information from multiple data sources and secondly to construct new, anonymous microdata with the appropriate variables and granularity. The synthetic data set must be realistic, i.e. statistically equivalent to the actual population of interest, and present the following characteristics (Münnich et al. 2003a, b):

- the distribution of the synthetic population by region and stratum must be quasi-identical to the distribution of the true population;
- marginal distributions and interactions between variables—the correlation structure of the true population—must be accurately represented;
- heterogeneities between subgroups, especially regional aspects, must be allowed;
- the records in the synthetic population should not be created by pure replication of units from the underlying sample, as this will usually lead to unrealistically small variability of units within smaller subgroups;
- data confidentiality must be ensured.

The idea of generating synthetic data is not new. Rubin (1993) suggested generating synthetic microdata using multiple imputation, and many other papers and books (see, e.g., Drechsler 2011) further developed these ideas.

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_6

In comparison to the previous (traditional) methods for data anonymization, we can show that the disclosure risk of synthetic data sets is very low and lower than for traditional methods, but typically synthetic data are less analytically valid than data sets modified by traditional anonymization methods. Thus, synthetic data sets are perfectly suited to act as public-use files for which researcher can test methods or for teaching purposes.

Synthetic data sets are not intended to replace traditional data sets for all research purposes, and will certainly not reduce the need to collect more and better data. But they are increasingly used for multiple practical applications. Synthetic data generation makes the dissemination and use of information contained in confidential data sets possible, by creating "replacement data sets" that can be shared as public use files for research or training.

Synthetic population data have further meanings und use compared to synthetic survey data only. Beside the more easy case of producing synthetic population data from populations (e.g. census data), we want to go a step further and produce synthetic populations from sample data. We point out advantages and reasons to simulate a synthetic population rather than a only synthetic sample.

- It is easier and theoretical sound to produce a whole population from survey's collected by complex sampling designs. The sampling weights can be considered adequately when blowing-up the data to a population. This cannot be fully considered by a model-based approach that only aims to produce a synthetic sample from a sample.
- We can also draw a survey sample from the simulated synthetic population using a different sampling design.
- Synthetic data generation also allows the creation of new, richer or "augmented" data sets that provide critical input for microsimulation (including spatial microsimulation) and agent-based modeling. Such data sets are particularly appealing for policymakers and development practitioners, who use them as input into simulation models for assessing the ex-ante distributional impact of policies and programs. Examples are found in multiple sectors, including health (Barrett et al. 2011; Brown and Harding 2002; Tomintz et al. 2008; Smith et al. 2011), transportation (Beckman et al. 1996; Barthelemy and Toint 2013), environment (Williamson et al. 2002), and others.
- Population data can be used for teaching purposes especially to show the effect of sampling designs on estimates.
- Design-based simulation studies needs population data since also the impact of the sampling design can be evaluated.

Multiple approaches have been proposed for the generation of synthetic population data, which can be classified into three broad categories: synthetic reconstruction, combinatorial optimization, and model-based generation of data.

For synthetic reconstruction methods, conditional probabilities are firstly either estimated from existing survey data or they can be possible also taken from other data sets or census's. In other words, the marginal distributions and conditional probabilities of relevant categorical socio-demographic variables covering the whole

population of interest is needed as a starting point. The synthetic population data set is typically generated sequentially, variable by variable. The values of a categorical variable are drawn based on conditional probabilities of a marginal distribution. A weakness of synthetic reconstruction methods is that synthetic individuals are generated by replicating individuals from the source microdata, i.e. it does not allow generation of combinations of variable categories that are not represented in the original sample data.

We do not point out more details about this method, since the model-based methods explained later in this chapter are more powerful, more flexible and more complex data sets can be produced, e.g., we can also simulate non-categorical variables and we can consider cluster structures or hierarchical structures in the data.

Combinatorial optimization methods are also used to simulate population data, but only for special tasks. Later on we use these kind of methods to calibrate populations. Typically, these methods are computational complex.

The model-based generation methods are best suited to produce synthetic data. We can consider cluster structures or hierarchical structures in the data and we can simulate any kind of variables and also combinations of values that are not exists in the sample.

6.2 Model-Based Generation of Synthetic Data

The model-based generation of synthetic data is a flexible and diverse approach. It consists of first deriving a model of the population from existing microdata and ancillary information, then "predicting" a synthetic population.

As already mentioned, to address the confidentiality problem connected with the release of publicly available microdata, Rubin (1993) proposed the generation of fully synthetic microdata sets using multiple imputation. The method is discussed in more detail by Raghunathan et al. (2003), Drechsler et al. (2008), Reiter (2009), Drechsler (2011). They mostly do not investigate the possible generation of structural zeros in combinations of variables as well as it is not easy or almost impossible to simulate complex data with cluster structures such as employees in enterprises or person in households nor it is easy to incooperate with the sampling design and sampling weights.

The generation of population microdata for selected surveys that form the foundation for Monte Carlo simulations is described by Münnich et al. (2003a, b). Their framework, however, was developed for household surveys with large sample sizes that primarily contain categorical variables. All steps of the procedure are performed separately for each stratum of the sampling design.

A different approach is proposed by Alfons et al. (2011), Templ and Filzmoser (2014), Templ et al. (2017). Their approach makes use of microdata from a representative sample of the population of interest, which is the only required input. In an initial step, the structure (e.g. household structure by age and sex, and possibly other key variables) is created. This is achieved by first estimating the number of

households of each size in the population of each stratum, taking into account the sample weights, then by randomly resampling the necessary number of households of each size from the sample. Additional categorical variables are then simulated using multi-nomial logistic regression models by random draws from observed conditional distributions within each combination of stratum, age or age group, and gender. Combinations that do not occur in the sample but are likely to occur in the true population can then be simulated. In the third step, continuous and semi-continuous variables are generated. One approach consists of imputing a discretized category of the continuous variable using multi-nomial logistic regression models, followed by random draws from uniform distributions within the imputed categories. For the largest categories, tail modeling based on generalized Pareto distribution can be performed. Another approach involves two-step regression models combined with random error terms. If necessary, the synthetic continuous variables can be split into components using an approach based on conditional resampling of fractions. For non-household data such, e.g., for employer-employee data, instead of households, the enterprise's structure is simulated equivalent to the approach of creating a household structure.

The schema presented in Fig. 6.1 represents a workflow to simulate a synthetic population (see also Templ et al. 2017). It is an adaptation, proposed by Alfons et al. (2011), of the framework of Münnich et al. (2003b). As a first step, sampling weights in the survey data are calibrated using the iterative proportional fitting technique to match population numbers in the census tables. The household structure of the synthetic population is then created by extrapolating the calibrated sample. Categorical and continuous variables are added to that structure using a modeling approach. Then the synthetic population is distributed among small(er) geographic areas. As a last step, the population data set is re-calibrated, this time using the a simulated annealing combinatorial optimization technique.

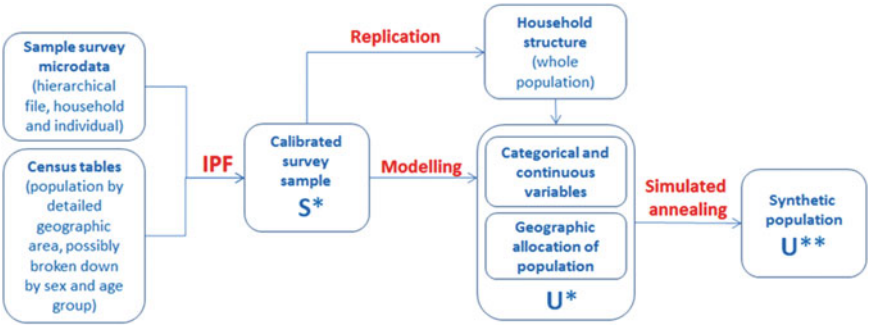


Fig. 6.1 Workflow for simulating the population. From a calibrated survey, a household structure for a whole population is simulated and then categorical or/and continuous variables are simulated. Optionally, combinatorial optimization methods allow to calibrate the simulated population to meet known population characteristics. Originally published in Templ et al. (2017). Published with kind permission of ©Matthias Templ, Bernhard Meindl, Alexander Kowarik and Oliver Dupriez 2016 under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium provided the original author(s) and source are credited

The proposed framework proceeds in a stepwise fashion, generating categorical variables first, then continuous variables. However, categorical and continuous variables can be simulated in any order, allowing for simulated continuous variables to be used as predictors for simulating categorical variables.

6.2.1 Setup of the Structure

This step is somehow completely different to the methods described in the next steps, since it does not contain a *response* versus *predictors* procedure. The setup of the structure (of households in socio-economic data, of enterprises in employee/employer data, ...) consists basically on replication of the existing structure of underlying data, by also considering the sampling weights (for simple random sampling, the weights are just $1/n$, with n the number of observations).

The structure is simulated separately for each combination of stratum k and household size l. First, the number of primary sampling units M_{kl} (e.g. households or enterprises) is estimated using the Horvitz-Thompson estimator (Horvitz and Thompson 1952):

$$\hat{M}_{kl} := \sum_{h \in H_{kl}^S} w_h, \tag{6.1}$$

where H_{kl}^S denotes the index set of primary sampling units in stratum k of the survey data with household size l, and $w_h, h \in H_{kl}^S$, are the corresponding primary sampling units weights. Similarly, let H_{kl}^U be the respective index set of primary sampling units in the population data such that $|H_{kl}^U| = \hat{M}_{kl}$. The wording *primary sampling units* is not very precise since we just mean those units corresponding the clusters. This is, e.g., households in socio-economic data or enterprises in business data. In the following we simply take the wording *clusters*, *households* or *enterprises*, i.e. if *household* is mentioned the corresponding formulas and methods also works equivalent or very similar for *enterprises*, as described in (Templ and Filzmoser 2014).

Basic information from the survey households/enterprises is resampled to prevent unrealistic structures in the population households/enterprises. Let x_{hij}^S and x_{hij}^U denote the value of person i from household h in variable j for the sample and population data, respectively, and let the first p_1 variables contain the basic information on the household structure. For each population household $h \in H_{kl}^U$, a survey household $h' \in H_{kl}^S$ is selected with probability $w_{h'}/\hat{M}_{kl}$ and the household structure is set to (see also Alfons et al. 2011)

$$x_{hij}^U := x_{h'ij}^S, \quad i = 1, \ldots, l, \ j = 1, \ldots, p_1. \tag{6.2}$$

For disclosure reasons, it is critical that only as few variables as possible should span the structure, since replication of units does not involve that the disclosure risk is low. Our suggestion is to use only age and gender information, which is typically

available in household surveys. For employee-employer data, for example, the size of the enterprise and the location of the enterprise might be used (for employee-employer data, have a look in Templ and Filzmoser 2014). Using only such few variables (e.g. less than four), the disclosure risk is very small, since a lot of equal combinations are produced for the synthetic population—and in addition, all other variables are fully synthetic by using modeling approaches.

6.2.2 Simulation of Categorical Variables

In contradiction to synthetic reconstruction approaches that estimates distributions directly by the corresponding relative frequency distributions in the underlying sample (such as used, for example, in Münnich et al. 2003a, b) we estimate conditional distributions with multinomial logistic regression models, regression trees (Hothorn et al. 2006a, b) or random forest (Breiman 2001) to overcome these shortcomings.

The basic procedure is as follows (Templ et al. 2017).

(i) The variable to be simulated (response) is selected from the sample **S**. The variables (including the household structure variables) used as predictors must be present in both the sample **S** and the population **U**. Other variables (rest) can be simulated afterwards.

$$sample \quad \boldsymbol{S} = \overbrace{\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & x_{1,j+1} & x_{1,j+2} & \cdots \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & x_{2,j+1} & x_{2,j+2} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,j} & x_{n,j+1} & x_{n,j+2} & \cdots \end{pmatrix}}^{\text{predictors} \quad \text{response} \quad \text{rest}}$$

(ii) The model matrix is built from the predictors available in the sample **S**. A multinomial logistic regression, a regression tree or conditional inference using random forests is modeled for the sample data with the variable to be simulated as response. This results to the fit of the regression coefficients $\boldsymbol{\beta}$ (or the regression tree in case of regression trees or random forests).

(iii) For every individual of the selected variable, predict the outcome category $\hat{x}_{i,j+1},\ i = 1, \ldots, N$ by using a multinomial regression model or the regression tree.

For each outcome $\hat{x}_{i,j+1},\ i = 1, ..., N$, its conditional probabilities for each R category is estimated by

$$\hat{p}_{i1} := \frac{1}{1 + \sum_{r=2}^{R} \exp(\hat{\beta}_{0r} + \hat{\beta}_{0r}\hat{x}_{i,1} + \ldots + \hat{\beta}_{jr}\hat{x}_{i,j})} \quad ,$$

$$\hat{p}_{ir} := \frac{\exp(\hat{\beta}_{0r} + \hat{\beta}_{0r}\hat{x}_{i,1} + \ldots + \hat{\beta}_{jr}\hat{x}_{i,j})}{1 + \sum_{r=2}^{R} \exp(\hat{\beta}_{0r} + \hat{\beta}_{0r}\hat{x}_{i,1} + \ldots + \hat{\beta}_{jr}\hat{x}_{i,j})} \quad ,$$

with $r = 2, \ldots R$ and $\hat{\beta}_{0r}, \ldots \hat{\beta}_{j-1,r}$ are the estimated coefficients from a multinomial model. The new values, $\hat{x}^*_{i,j+1}$, are drawn given these probabilities. Schematically, this is:

$$
\begin{array}{l}
\\
\textit{population} \quad \boldsymbol{U} = \\
\\
\\
\\
\\
\end{array}
\overbrace{
\begin{array}{cccc|c}
\multicolumn{4}{c|}{\hat{\boldsymbol{\beta}} \times \text{ pred.} \approx} & \hat{\mathbf{x}}^*_{j+1} \\
\hat{x}_{1,1} & \hat{x}_{1,2} & \cdots & \hat{x}_{1,j} & \hat{x}^*_{1,j+1} \\
\hat{x}_{2,1} & \hat{x}_{2,2} & \cdots & \hat{x}_{2,j} & \hat{x}^*_{1,j+1} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\hat{x}_{N,1} & \hat{x}_{N,2} & \cdots & \hat{x}_{N,j} & \hat{x}^*_{1,j+1}
\end{array}
}
$$

This scheme is in the following also described more precisely. Let $\boldsymbol{x}_j^S = (x_{1j}^S, \ldots, x_{nj}^S)'$ and $\boldsymbol{x}_j^U = (x_{1j}^U, \ldots, x_{Nj}^U)'$ denote the variables in the sample and population, respectively, where n and N give the corresponding number of individuals. The additional categorical variables are thereby given by the indices $p_1 < j \leq p_2$. Furthermore, the personal sample weights are denoted by $\boldsymbol{w} = (w_1, \ldots, w_n)'$. Either multinomial logistic regression models (see, e.g., Simonoff 2003), regression trees (Hothorn et al. 2006a, b) or random forest (Breiman 2001) are fitted for each stratum separately.

The following procedure is performed for each stratum k and each variable to be simulated, given by the index j, $p_1 < j \leq p_2$. As mentioned above, the basic procedure works sequentially, where newly simulated variables serve as predictors for simulated a further variable. Let I_k^S and I_k^U be the index sets of individuals in stratum k for the survey and population data, respectively. The survey data given by the indices in I_k^S is used to fit the model with response $\boldsymbol{x}_j^S$ and predictors $\boldsymbol{x}_1^S, \ldots, \boldsymbol{x}_{j-1}^S$, thereby considering the sample weights w_i, $i \in I_k^S$. Furthermore, let $\{1, \ldots, R\}$ be the set of possible outcome categories of the response variable. In particular, the number of possible outcomes is denoted by R. For every individual $i \in I_k^U$, the conditional probabilities $p_{ir}^U := P(x_{ij}^U = r | x_{i1}^U, \ldots, x_{i,j-1}^U)$ are estimated by

$$
\begin{aligned}
\hat{p}_{i1}^U &:= \frac{1}{1 + \sum_{r=2}^R \exp(\hat{\beta}_{0r} + \hat{\beta}_{1r} x_{i1}^U + \ldots + \hat{\beta}_{j-1,r} x_{i,j-1}^U)}, \\
\hat{p}_{ir}^U &:= \frac{\exp(\hat{\beta}_{0r} + \hat{\beta}_{1r} x_{i1}^U + \ldots + \hat{\beta}_{j-1,r} x_{i,j-1}^U)}{1 + \sum_{r=2}^R \exp(\hat{\beta}_{0r} + \hat{\beta}_{1r} x_{i1}^U + \ldots + \hat{\beta}_{j-1,r} x_{i,j-1}^U)}, \quad r = 2, \ldots, R,
\end{aligned}
\tag{6.3}
$$

where $\hat{\beta}_{0r}, \ldots, \hat{\beta}_{j-1,r}$, $r = 2, \ldots, R$, are the estimated coefficients (see, e.g., Simonoff 2003). The values of $\boldsymbol{x}_j^U$ for the individuals $i \in I_k^U$ are then drawn from the corresponding conditional distributions.

Note that for simulating the jth variable, $p_1 < j \leq p_2$, the $j - 1$ previous variables are used as predictors for the default approach. This means that the order of the additional categorical variables may be relevant. However, once such a variable is generated in the population, that information should certainly be used for

simulating the remaining variables. In general, changing the order of the variables did not produce significantly different results (Alfons et al. 2011). Alternatively, the procedure could be continued iteratively once all additional variables are available in the population, in each step using all other variables as predictors. Nevertheless, such a procedure would be computationally very expensive for real-life sized population data. It is thus often of practical use to not use all available variables, i.e. the variables from the survey sample that are already produced for the population, but only a subset. We report further on this issue in Chap. 8 when producing a synthetic data set in Sect. 8.7.

Estimating the conditional distributions with multinomial logistic regression models allows to simulate combinations that do not occur in the sample but are likely to occur in the true population. Such combinations are called *random zeros*, as opposed to *structural zeros*, which are impossible to occur (e.g., Simonoff 2003). For close-to-reality populations, such structural zeros need to be reflected. This can be done by setting $p^U_{ir'} := 0$, where r' is an impossible value for x_{ij} given $x_{i1}, \ldots, x_{i,j-1}$, and adjusting the other probabilities so that $\sum_{r=1}^{R} p^U_{ir} = 1$.

Keep in mind that the idea of the proposed data simulation framework is to proceed in a stepwise fashion, generating different types of variables in each step and starting with the categorical ones. However, the procedure could easily be modified to allow for previously simulated continuous predictors when simulating a categorical variable.

6.2.3 Simulation of Continuous Variables

We favor three approaches to simulate continuous variables. The first involves fitting a multinomial model and random drawing from the resulting categories. This approach is equivalent to the simulation of categorical variables described in the previous section, whereas first the continuous variable is categorized and the multinomial model is estimated. In the last step, random draws using a random number generator are taken from the intervals of the categories into which predictions at population level fall. In that case, values from the largest categories could be drawn from a generalized Pareto distribution, since the values are not expected to be uniformly distributed.

The second approach is based on a two-step regression model with random error terms. In case of semi-continuous variables, a logistic regression is applied in a first step. In the second step, a linear regression is applied only to observations for which the prediction of the logistic regression is closer to one than to zero. This is necessary when considering semi-continuous distributions, otherwise only the linear regression imputation is applied. A random error value is added to avoid all individuals with the same set of predictors receiving the same value for the predicted variables, which would underestimate the variance. Random draws can be based on normal assumption

or, preferably, taken from the regression residuals. The first step can be omitted when simulating continuous variables (and not semi-continuous ones).

The third approach is for count variables, just a Poisson model is used for the link function in a general linear model, instead of a normal distribution.

Let us continue with the concept and notation from the previous section, and let $\boldsymbol{x}_j^S$ and $\boldsymbol{x}_j^U$, $p_2 < j \leq p_3$ denote the continuous variables. Two different approaches are presented in the following. Both are able to handle semi-continuous variables, i.e. variables that contain a large amount of zeros.

6.2.3.1 Multinomial Model with Random Draws from Resulting Categories

This approach is based on the simulation of categorical variables described in the previous section, since a continuous variables is cut into ordered categories first. The following steps are performed for each variable to be simulated, given by the index j, $p_2 < j \leq p_3$. First, the variable $\boldsymbol{x}_j^S$ is divided into categories. This is done in a different manner for continuous and semi-continuous variables. For continuous variables, $R+1$ breakpoints $b_1 < \ldots < b_{R+1}$ are used to define the discretized variable $\boldsymbol{y}^S = (y_1^S, \ldots, y_n^S)'$ as (see also Alfons et al. 2011)

$$y_i^S := \begin{cases} 1 & \text{if } b_1 \leq x_{ij}^S \leq b_2, \\ r & \text{if } b_r < x_{ij}^S \leq b_{r+1},\ r = 2, \ldots, R. \end{cases} \tag{6.4}$$

For semi-continuous variables, zero is a category of its own, and breakpoints for negative and positive values are distinguished. Let $b_{R^-+1}^- < \ldots < b_1^- = 0 = b_1^+ < \ldots < b_{R^++1}^-$ be the breakpoints. Then $\boldsymbol{y}^S$ is defined as

$$y_i^S := \begin{cases} -r & \text{if } R^- > 0 \text{ and } b_{r+1}^- \leq x_{ij}^S < b_r^-,\ r = R^-, \ldots, 1, \\ 0 & \text{if } x_{ij}^S = 0, \\ r & \text{if } R^+ > 0 \text{ and } b_r^+ < x_{ij}^S \leq b_{r+1}^+,\ r = 1, \ldots, R^+. \end{cases} \tag{6.5}$$

Note that the cases of only non-negative or non-positive values in $\boldsymbol{x}_j^S$ are considered in (6.5).

Multinomial logistic regression models with response $\boldsymbol{y}^S$ and predictors $\boldsymbol{x}_1^S$, …, $\boldsymbol{x}_{j-1}^S$ are then fitted for every stratum k separately, as described in the previous section, in order to simulate the values of the categorized population variable $\boldsymbol{y}^U = (y_1^U, \ldots, y_N^U)'$.

Finally, the values of $\boldsymbol{x}_j^U$ are generated by random draws from uniform distributions within the corresponding categories of $\boldsymbol{y}^U$. For continuous variables, the values of individual $i = 1, \ldots, N$ are generated as

$$x_{ij}^U \sim U(b_r, b_{r+1}) \text{ if } y_i^U = r. \tag{6.6}$$

For semi-continuous variables, the values of individual $i = 1, \ldots, N$ are set to $x_{ij}^U := 0$ if $y_i^U = 0$, while the non-zero observations are generated as

$$x_{ij}^U \sim \begin{cases} U(b_{r+1}^-, b_r^-) & \text{if } y_i^U = -r < 0, \\ U(b_r^+, b_{r+1}^+) & \text{if } y_i^U = r > 0. \end{cases} \tag{6.7}$$

The idea behind this approach is to divide the data into relatively small subsets. If the intervals are too large, using uniform distributions may be an oversimplification and will lead to poor results. It should be rather ensured that the breakpoints for the discretization are chosen in such a way that the empirical distribution is well reflected in the simulated population variable. However, the higher the number of breakpoints the longer the computation time. As a guideline we propose quantiles in steps of 10% and 1, 5, 95 and 99% quantiles in the tails of the distribution.

When simulating variables that contain extreme values, such as income, *tail modeling* should be considered. In that case, values from the largest categories could be drawn from a generalized Pareto distribution (GPD). The cumulative distribution function of the GPD is defined as

$$F_{\mu,\sigma,\xi}(x) = \begin{cases} 1 - \left(1 + \dfrac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}}, & \xi \neq 0, \\ 1 - \exp\left(-\dfrac{x-\mu}{\sigma}\right), & \xi = 0, \end{cases}$$

where μ is the location parameter, $\sigma > 0$ is the scale parameter and ξ is the shape parameter. The range of x is $x \geq 0$ when $\xi \geq 0$ and $\mu \leq x \leq \mu - \frac{\sigma}{\xi}$ when $\xi < 0$. See, e.g., Embrechts et al. (1997) for details on the *peaks over threshold* approach for fitting the GPD. Note that other distributions may be used for tail modeling as well (see, e.g., Kleiber and Kotz 2003), and outliers might be replaced beforehand using semi-parametric tail modelling for complex survey samples (Alfons et al. 2013).

6.2.3.2 (Two-Step) Regression Model with Random Error Terms

The second approach is based on linear regression combined with random error terms. Note that adding random errors is quite similar as to draw from predictive distributions typically done for classical multiple imputation approaches. A classical model can be used for continuous data, but not for semi-continuous variables Semi-continuous variables must be simulated using a two-step model. A log-linear model is used to predict whether a value of an observation is zero or not. Then a linear model is used to predict the non-zero values (Alfons et al. 2011).

The following procedure is repeated for each variable to be simulated, given by the index j, $p_2 < j \leq p_3$.

For semi-continuous variables, the first step is to simulate whether x_{ij}^U, $i = 1, \ldots, N$, is zero or not. This is done by fitting logistic regression models (see, e.g., Simonoff 2003) for each stratum separately. The binary response variable $\boldsymbol{y}^S = (y_1^S, \ldots, y_n^S)'$ is defined as

$$y_i^S := \begin{cases} 0 & \text{if } x_{ij} = 0, \\ 1 & \text{else.} \end{cases} \tag{6.8}$$

For each stratum k, the observations given by the index set I_k^S are used to fit the model with response $\boldsymbol{y}^S$ and predictors $\boldsymbol{x}_1^S, \ldots, \boldsymbol{x}_{j-1}^S$. The sample weights $w_i, i \in I_k^S$, are considered in the model fitting process by using a weighted maximum likelihood approach. For every individual $i \in I_k^U$, the conditional probabilities $p_i^U := P(y_i^U = 1 | x_{i1}^U, \ldots, x_{i,j-1}^U)$ that x_{ij}^U is non-zero are estimated by

$$\hat{p}_i^U := \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_{i1}^U + \ldots + \hat{\beta}_{j-1} x_{i,j-1}^U)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x_{i1}^U + \ldots + \hat{\beta}_{j-1} x_{i,j-1}^U)}, \tag{6.9}$$

where $\hat{\beta}_0, \ldots, \hat{\beta}_{j-1}$ are the estimated coefficients (e.g., Simonoff 2003). The values y_i^U, $i \in I_k^U$, are then drawn from the corresponding conditional distributions. Consequently, the zeros in the simulated semi-continuous variable are given by $x_{ij}^U := 0$ if $y_i^U = 0$. For the second step, the non-zero observations are indicated by $\tilde{I}_k^S := \{i \in I_k^S : y_i^S = 1\}$ and $\tilde{I}_k^U := \{i \in I_k^U : y_i^U = 1\}$ (See also Alfons et al. 2011).

For continuous variables, on the other hand, $\tilde{I}_k^S := I_k^S$ and $\tilde{I}_k^U := I_k^U$ are used in the following. Linear regression models are fitted for every stratum separately. In order to obtain more robust models, trimming parameters α_1 and α_2 are introduced. The following procedure is carried out for each stratum k. Let the observations to be used for fitting the model be given by the index set $I_{\alpha_1}^{\alpha_2} := \{i \in \tilde{I}_k^S : q_{\alpha_1} < x_{ij} < q_{1-\alpha_2}\}$, where q_{α_1} and $q_{1-\alpha_2}$ are the corresponding α_1 and $1 - \alpha_2$ quantiles, respectively. The linear model is then given by

$$x_{ij}^S = \beta_0 + \beta_1 x_{i1}^S + \ldots + \beta_{j-1} x_{i,j-1}^S + \varepsilon_i^S, \quad i \in I_{\alpha_1}^{\alpha_2}, \tag{6.10}$$

where ε_i^S are random error terms. Using the weighted least squares approach with weights $w_i, i \in I_{\alpha_1}^{\alpha_2}$, coefficients $\hat{\beta}_0, \ldots, \hat{\beta}_{j-1}$ are obtained (see, e.g., Weisberg 2005) and the population values are estimated by

$$\hat{x}_{ij}^U = \hat{\beta}_0 + \hat{\beta}_1 x_{i1}^U + \ldots + \hat{\beta}_{j-1} x_{i,j-1}^U + \varepsilon_i^U, \quad i \in \tilde{I}_k^U. \tag{6.11}$$

The random error terms ε_i^U need to be added since otherwise individuals with the same set of predictor values would receive the same value in x_{ij}^U. There are two suggestions on how to generate the random error terms (Alfons et al. 2011):

- random draws from the residuals

$$\hat{r}_i^S = x_{ij}^S - \left(\hat{\beta}_0 + \hat{\beta}_1 x_{i1}^S + \ldots + \hat{\beta}_{j-1} x_{i,j-1}^S\right), \quad i \in I_{\alpha_1}^{\alpha_2}. \tag{6.12}$$

- random draws from a normal distribution $\mathcal{N}(\mu, \sigma^2)$. The parameters μ and σ are thereby estimated robustly with median and MAD, respectively.

The first approach is more data-driven, while the second approach is in accordance with the theoretical assumption of normally distributed errors. For both, the trimming parameters α_1 and α_2 need to be selected carefully. These trimming parameters/thresholds are useful to deal with residual outliers. If these parameters are too small, very large random error terms due to outliers may result in large deviations especially in the tails of the distribution. If they are too large, the random error terms may not introduce enough variability.

For variables such as income, a log-transformation may be beneficial before fitting the linear model. Equation (6.25) is then changed to

$$\log x_{ij}^S = \beta_0 + \beta_1 x_{i1}^S + \ldots + \beta_{j-1} x_{i,j-1}^S + \varepsilon_i^S, \quad i \in I_{\alpha_1}^{\alpha_2}. \tag{6.13}$$

In that case, the population values are estimated by

$$\hat{x}_{ij}^U = \exp(\hat{\beta}_0 + \hat{\beta}_1 x_{i1}^U + \ldots + \hat{\beta}_{j-1} x_{i,j-1}^U + \varepsilon_i^U), \quad i \in \tilde{I}_k^U. \tag{6.14}$$

However, the log-transformation causes problems with negative values, which is realistic for income A simple remedy is of course to add a constant $c > 0$ to x_{ij}^S to obtain positive values, i.e. to use $\log(x_{ij}^S + c)$ in the left-hand side of Eq. (6.13). This constant then needs to be subtracted from the right hand side of Eq. (6.14). This is e.g. also implemented in the R package **simPop** (Templ et al. 2017). Another possibility is to combine the two presented approaches for simulating (semi-)continuous variables. A multinomial model with one category for positive values and certain categories for non-positive values is applied in the first step. Positive values are then simulated using a linear model for the log-transformed data, while negative values are drawn from uniform distributions within the respective simulated categories (Alfons et al. 2011).

6.2.4 *Splitting Continuous Variables into Components*

If continuous variables are components, e.g. income components, then the following problems need to be considered (cf. Kraft 2009). Even for a moderate number of components, it may be too complex to consider all the dependencies between the components and the other variables, as well as between the components themselves. Moreover, also various components are very sparse, e.g. particular income components typically contain only few non-zero observations. To manage these problems, a simple but effective approach based on conditional resampling of fractions has

been developed by Alfons et al. (2011). Only very few highly influential categorical variables should thereby be considered for conditioning.

Resampling fractions has the advantage that it avoids unrealistic or unreasonable combinations in the simulated components. At the same time, it does not result in pure replication, as the absolute values for simulated individuals are in general quite different from the corresponding individuals in the underlying survey data.

Let us take the description of the resampling fractions approach from Alfons et al. (2011), and let $\boldsymbol{z}^S = (z_1^S, \ldots, z_n^S)'$ and $\boldsymbol{z}^U = (z_1^U, \ldots, z_N^U)'$ denote the variable giving the total in the sample and population, respectively, and let $\boldsymbol{x}_j^S$ and $\boldsymbol{x}_j^U$, $p_3 < j \leq p_4$, denote the variables containing the components. First, the fractions of the components with respect to the total are computed for the sample:

$$y_{ij}^S := \frac{x_{i,p_3+j}^S}{z_i^S}, \quad i \in I_r^S, \; j = 1, \ldots, p_4 - p_3. \tag{6.15}$$

For the second step, let J_c be the index set of the conditioning variables. This step is performed separately for every combination of outcomes $\boldsymbol{r} = (r_j)_{j \in J_c}$. Let $I_r^S := \{i : x_{ij}^S = r_j \, \forall j \in J_c\}$ and $I_r^U := \{i : x_{ij}^U = r_j \, \forall j \in J_c\}$ be the index sets of individuals in the survey and population data, respectively, with the corresponding outcomes in the conditioning variables. For each individual $i \in I_r^U$ in the population, an individual $i' \in I_r^S$ from the survey data is selected with probability $w_{i'} / \sum_{i \in I_r^S} w_i$ and the values of the components are set to

$$x_{i,p_3+j}^U := z_i^U y_{i'j}^S, \quad j = 1, \ldots, p_4 - p_3. \tag{6.16}$$

If no observations for combination $\boldsymbol{r}$ exist in the sample, i.e. if $I_r^S = \emptyset$, a suitable donor $\boldsymbol{r}'$ is selected by minimizing a suitable distance measure such as the Manhattan distance $d_1(\boldsymbol{r}, \boldsymbol{s}) = \|\boldsymbol{r} - \boldsymbol{s}\|_1$. Then $I_r^S := I_{r'}^S$ is used in the above steps (Alfons et al. 2011).

6.3 Disclosure Risk of Synthetic Data

The motivation for generating close-to-reality population data is to make the resulting data sets publicly available to researchers for use in simulation studies or courses on survey statistics. Therefore, the disclosure risk of such data needs to be very low, while at the same time the multivariate structure should be as realistic as possible.

For the production of synthetic data it is favorable to use information of the raw non-disclosed microdata to generate synthetic populations, since the quality of the population data is increased by using unperturbed sample data as basis to generate population data as when using already perturbed data. Especially, when anonymizing data using global recoding, less categories result and additionally missing values are present after local suppression. Thus we only discuss the disclosure risk from synthetic data that are produced from raw microdata.

Generally, it always depends on the attacker's available information. If an attacker has no additional information than the synthetic data, he might be always unsure if a re-identification gives valuable information on (sensitive) variables of interest. Technically speaking, he will do *penetration tests* without having additional data sources available. This means that he tries to learn from the data, looking at unique patterns, etc. If he has also additional information, e.g. on other data bases, he will try to match information from the synthetic data and his data bases, and with a successful match he would potentially got additional information on some variables that he do not observe with his data bases only. However, for synthetic data it does not mean that even a correct match of key variables the other information on sensitive variables are real. Not at all. They are typically produced by random draws from certain intervals and probabilities. He can only learn by chance, and he cannot be sure if the information is realistically close to the truth. Next he can do target matching by only looking at certain rare frequencies of key variables. A typical example is to look at the largest household(s) in the synthetic data, and match this household(s) with his data bases. Even in the case that only synthetic survey data are produced, it is likely to produce a different amount of largest household by random draws from probabilities. It can likely be happen that, say, a 10 persons household is present in the raw data but not or multiple times in the synthetic data, with even randomly drawn values on sensitive variables. And this is even more safe when simulating synthetic population data. In case for extreme households, for example, re-identification might be possible but no additional information is gained for an intruder. By producing the household population structure on age, gender and region we produce in the average N/n nine-person households in the population. In case of the Austrian EU-SILC data this would be about 550 9-person households. In addition, every value on categorical variable of these households are drawn by a probability mechanism based on surely not perfect models. Furthermore, the continuous variables are also drawn in an prediction interval. In the end we produce a lot of 9-person households where some of them might have the same values on key variables just by chance, but they all have different values on sensitive variables. Looking on such extreme households or values on persons, we easily can see that an intruder can hardly disclose information on real persons.

In the following we concentrate on matching and consider some simplified versions of matching that all can be seen as worst case scenarios. We will see that even for worst case scenarios, the disclosure risk is zero or approx. zero.

A popular global measure of the re-identification risk for survey data is given by the number of uniqueness's in the sample that are unique in the population as well (see Sect. 3.9).

Let us partly write down again the formulas to describe our risk measure in detail. Let m categorical key variables in the sample and population data be denoted by $\boldsymbol{x}_j^S = (x_{1j}^S, \ldots, x_{nj}^S)'$ and $\boldsymbol{x}_j^P = (x_{1j}^P, \ldots, x_{Nj}^P)'$, respectively, $j = 1, \ldots, m$, where n and N give the corresponding number of observations. For an observation in the sample given by the index $c = 1, \ldots, n$, let J_c^S and J_c^P denote the index sets of observations in the sample and population data, respectively, with equal values in the m key variables (cf Templ and Alfons 2010):

$$\begin{aligned} J_c^S &:= \{j = 1, \ldots, n : x_{jk}^S = x_{ck}^S,\ k = 1, \ldots, m\}, \\ J_c^P &:= \{j = 1, \ldots, N : x_{jk}^P = x_{ck}^S,\ k = 1, \ldots, m\}. \end{aligned} \tag{6.17}$$

Furthermore, we define an indicator function $\mathcal{I}$ as

$$\mathcal{I}(J) := \begin{cases} 1 & \text{if } |J| = 1, \\ 0 & \text{else.} \end{cases} \tag{6.18}$$

The global disclosure risk measure can then be expressed by

$$\tau_1 := \sum_{c=1}^{n} \mathcal{I}(J_c^S) \cdot \mathcal{I}(J_c^P). \tag{6.19}$$

Note that the notation in (6.19) differs slightly from the common definition.

Clearly, the lower the risk of re-identification, the higher the corresponding population frequency count. If the population frequency count is sufficiently high, it is not possible for an intruder to assign the observation for which they hold information with absolute certainty. Hence the intruder does not know whether the re-identification was successful. However, the true frequency counts of the population are usually unknown and need to be estimated by modeling the distribution of the population frequencies.

6.3.1 Confidentiality of Synthetic Population Data

The global disclosure risk measure τ_1 is now modified to estimate the disclosure risk for synthetic population data in certain scenarios instead of survey data.

Based on the ideas proposed in Rubin (1993), Little (1993), the generation of *fully* or *partially* synthetic population data using multiple imputation is discussed in Raghunathan et al. (2003), Drechsler et al. (2008), Reiter (2009). More precisely, let p be the number of variables in the sample and let the first k, $1 \leq k < p$, categorical variables be available for the population data from administrative sources. These first k variables are released unchanged, while the remaining $p - k$ variables are estimated using regression based multiple imputation. It is important to note that the first k variables of real population data may still contain unique combinations after cross tabulation, therefore further investigation may be necessary to ensure confidentiality. Probabilities of re-identification for such synthetic data have been studied in Reiter (2009), based on the work of Duncan and Lambert (1986), Fienberg et al. (1997), by matching the synthetic data with the intruder's data on some pre-defined key variables.

The situation for synthetic population data generated by the approach presented in this chapter (and here: Alfons et al. 2011; Templ and Filzmoser 2014; Templ et al. 2017) is different. A very low number of basic categorical variables are generated in

the first step by resampling from the actual survey data. Since the sample weights are thereby used as probability weights, on average k-anonymity is provided with respect to these basic variables, where k denotes the smallest sample weight. In surveys, k is typically very high (>500), hence the disclosure risk is very low. However, additional categorical and continuous variables are generated based on models obtained from the actual survey data. In particular, the generation of continuous variables involves random draws from certain intervals.

With the additional categorical variables, some unique combinations may be introduced in the synthetic population data. If such a combination is not unique in the real population, it is not useful to an intruder. On the other hand, if such a combination is unique in the real population as well, it must be ensured that the values of the other variables in the synthetic population data are not too close to the real values. Most notably, it is of interest to measure the difference in continuous variables of the successfully matched statistical units.

In addition, unique combinations in the real population may even be critical if they are not unique in the synthetic population data. An intruder could in this case look for all occurrences of such a combination in the synthetic population. If the corresponding units have too similar values in a (continuous) variable of interest, the intruder may be able to infer some information on the original value, since the synthetic values have been predicted with models obtained from the real sample data.

To summarize, for synthetic data simulated with the presented model-based approach confidentiality is given by

- populations are produced, and we start from about 2–3 structural variables that are resampled. The frequency counts are very large for the combination of these variables;
- additional variables are simulated using random draws from given probabilities. It is possible even unlikely that k-anonymity is violated, but even if this is the case an intruder will not gain additional information since all categorical variables are simulated using random draws;
- further continuous variables are simulated by random draws. Even in the rare case that an individual have the same values on the combination of the key variables, it is very unlikely that the values on sensitive variables are the same as in reality, i.e. the probability is very high that they differ from reality.

In order to investigate these issues in more detail, various disclosure scenarios are introduced in the following section.

6.3.2 Disclosure Scenarios for Synthetic Population Data

Five different scenarios are considered to evaluate the confidentiality of synthetic data generated with the framework proposed in this chapter. These scenarios are motivated by the synthetic EU-SILC population data, hence only a continuous variable is considered to contain confidential information, while there are m categorical key

variables. We also define only worst case scenarios and keep it simple by excluding neighborhood disclosure (Rinott and Shlomo 2007).

In the case of EU-SILC, we may choose a disclosure scenario, i.e. to define the key variables. The confidential, sensitive variable might be *personal net income* and the key variables are *region*, *household size*, *age category*, *gender*, *economic status* and *citizenship*. Let the confidential continuous variable for the original sample and synthetic population, respectively, be denoted by $\boldsymbol{y}^S = (y_1^S, \ldots, y_n^S)'$ and $\boldsymbol{y}^U = (y_1^U, \ldots, y_N^U)'$, while the categorical key variables are denoted by $\boldsymbol{x}_j^S = (x_{1j}^S, \ldots, x_{nj}^S)'$ and $\boldsymbol{x}_j^U = (x_{1j}^U, \ldots, x_{Nj}^U)'$, $j = 1, \ldots, m$, analogous to the definitions in Sect. 3. Furthermore, let J_c^S be defined as in (6.17) and let J_c^U be defined accordingly as

$$J_c^U := \{j = 1, \ldots, N : x_{jk}^U = x_{ck}^S,\ k = 1, \ldots, m\}. \tag{6.20}$$

In the following scenarios, the intruder has knowledge of the m key variables for all observations from the original sample and tries to acquire information on the confidential variable.

It should be noted that the link to the global risk measure from (6.19) is loosened in the following. Disclosure is considered to occur if the value of the confidential variable for a unique combination of key variables in the sample can be closely estimated from the synthetic population data, given a pre-specified value of accuracy p. However, such a sample uniqueness does not need to be unique in the true population, in which case close estimation of the confidential variable would not necessarily result in disclosure. In this sense, the following scenarios can be considered worst case scenarios and the reidentification risk is thus overestimated. Proper analysis with estimation of the true population uniqueness's is future research. Important for this book is that the readers should give an impression that the disclosure risk of synthetic data converge to zero. We list five scenarios from Templ and Alfons (2010).

Scenario 1: Attack Using One-to-One Matches in Key Variables with Information on the Data Generation Process.

The intruder in this scenario tries to find on-to-one matches between their data and the synthetic population data. Moreover, they know the intervals from which the synthetic values were drawn. Let these intervals be denoted by $[l_j, u_j]$, $j = 1, \ldots, N$, and let l be a function giving the length of an interval defined as $l([a, b]) := b - a$ and $l(\emptyset) := 0$. With a pre-specified value of accuracy p defining a symmetric interval around a confidential value, (6.19) is reformulated as

$$\tau_1 := \sum_{c=1}^{n} \mathcal{I}(J_c^S) \cdot \mathcal{I}(J_c^U) \cdot \frac{l([y_c^S(1-p), y_c^S(1+p)] \cap [l_{j_c}, u_{j_c}])}{l([l_{j_c}, u_{j_c}])}, \tag{6.21}$$

where $j_c \in J_c^U$ if $|J_c^U| = 1$, i.e. j_c is the index of the unit in the synthetic population with the same values in the key variables as the cth unit in the intruder's data if such a one-to-one match exists, otherwise it is a dummy index. The last term in (6.21)

thereby gives the probability that for the successfully matched unit, the synthetic value drawn from the interval $[l_{j_c}, u_{j_c}]$ is sufficiently close to the original value y_c^S.

Scenario 2: Attack Using One-to-One Matches in Key Variables without Information on the Data Generation Process.

In general, an intruder does not have any knowledge on the intervals from which the synthetic values were drawn. In this case, re-identification is successful if the synthetic value itself of a successfully matched unit is sufficiently close to the original value. The risk of re-identification thus needs to be reformulated as

$$\tau_2 := \sum_{c=1}^{n} \mathcal{I}(J_c^S) \cdot \mathcal{I}(J_c^U) \cdot \mathbb{I}_{[y_c^S(1-p), y_c^S(1+p)]}(y_{j_c}^U), \tag{6.22}$$

where j_c is defined as above and $\mathbb{I}_A$ denotes the indicator function for a set A.

Scenario 3: Attack Using All Occurrences in Key Variables with Information on the Data Generation Process.

This scenario is an extension of Scenario 1, in which the intruder does not only try to find one-to-one matches, but looks for all occurrences of a unique combination from their data in the synthetic population data. Keep in mind that the intruder in this scenario knows the intervals from which the synthetic values were drawn. For a unique combination in the intruder's data, re-identification is possible if the probability that the synthetic values of all matched units are sufficiently close to the original value. Hence the disclosure risk from (6.21) changes to

$$\tau_3 := \sum_{c=1}^{n} \mathcal{I}(J_c^S) \cdot \prod_{j \in J_c^U} \frac{l([y_c^S(1-p), y_c^S(1+p)] \cap [l_j, u_j])}{l([l_j, u_j])}. \tag{6.23}$$

Scenario 4: Attack Using All Occurrences in Key Variables without Information on the Data Generation Process.

In an analogous extension of Scenario 2, re-identification of a unique combination from the intruder's data is successful if the synthetic values themselves of all matched units are sufficiently close to the original value. Equation (6.22) is in this case rewritten as

$$\tau_4 := \sum_{c=1}^{n} \mathcal{I}(J_c^S) \cdot \prod_{j \in J_c^U} \mathbb{I}_{[y_c^S(1-p), y_c^S(1+p)]}(y_j^U). \tag{6.24}$$

Scenario 5: Attack Using Key Variables for Model Predictions.

In this scenario, the intruder uses the information from the synthetic population data to obtain a linear model for $\boldsymbol{y}^U$ with predictors $\boldsymbol{x}_j^U$, $j = 1, \ldots, m$:

$$\boldsymbol{y}^U = \beta_0 + \beta_1 \boldsymbol{x}_1^U + \ldots + \beta_m \boldsymbol{x}_m^U + \varepsilon. \tag{6.25}$$

For a unique combination of the key variables, re-identification is possible if the corresponding predicted value is sufficiently close to the original value. Let the predicted values of the intruder's data be denoted by $\hat{\boldsymbol{y}}^S = (\hat{y}_1^S, \ldots, \hat{y}_n^S)'$. Then the disclosure risk can be formulated as

$$\tau_5 := \sum_{c=1}^{n} \mathcal{I}(J_c^S) \cdot \mathbb{I}_{[y_c^S(1-p), y_c^S(1+p)]}(\hat{y}_c^S). \tag{6.26}$$

Note that for large population data, the computational costs for fitting such a regression model are very high, so an intruder needs to have a powerful computer with very large memory. Furthermore, the intruder could also perform a stepwise model search using an optimality criterion such as the AIC Akaike (1970).

The disclosure risk of the synthetic Austrian EU-SILC population data described in Sect. 8.7 is analyzed in the following with respect to the scenarios defined in the previous section. In these scenarios, the intruder has knowledge of the categorical key variables *region*, *household size*, *age category*, *gender*, *economic status* and *citizenship* for all observations in the original sample that was used to generate the data. In addition, the intruder tries to obtain information on the confidential variable *personal net income*. The original sample thereby consists of $n = 14\,883$ and the synthetic population and $N = 8\,182\,218$ observations.

Note that this chapter only evaluates the risk of re-identification for this specific synthetic data set. In order to get more general results regarding confidentiality of the data generation process, many data sets need to be generated in a simulation study and the average values need to be reported. This task, however, is beyond the scope of this book.

Table 6.1 lists the results for the risk measures for the investigated scenarios using different values of the accuracy parameter p (see also Templ and Alfons 2010). Besides the absolute values, the relative values with respect to the size of the intruder's data set are presented, which give the probabilities of successful re-identification.

The results show that even if an intruder is able to re-identify an observation, they do not gain any useful information, as the probability that the obtained value is sufficiently close to the original value is extremely low.

In particular if the intruder tries to find one-to-one matches (Scenarios 1 and 2), the probability of a successful re-identification is only positive for $p = 0.05$ and if they have information on the data generation process, i.e. the intervals from which the synthetic values were drawn.

If the intruder looks for all occurrences of a unique combination from their data in the synthetic population, using information on the data generation process hardly changes the probabilities of re-identification (Scenario 3). This is not a surprise given the formula in (6.23), since for such a unique combination, the probabilities that the corresponding synthetic values are sufficiently close to the original value need to be multiplied. On the other hand, if the intruder uses only the synthetic values (Scenario 4), some observations are successfully re-identified. Nevertheless, the probabilities of re-identification are extremely low.

Table 6.1 Results for Scenarios 1–5 using different values for the accuracy parameter p

Scenario	Risk measure	p		
		0.01	0.02	0.05
1	τ_1	0	0	0.052
2	τ_2	0	0	0
3	τ_3	$1.1 \cdot 10^{-8}$	$1.2 \cdot 10^{-6}$	0.053
4	τ_4	15	15	15
5	τ_5	20	43	110
1	τ_1/n	0	0	$3.5 \cdot 10^{-6}$
2	τ_2/n	0	0	0
3	τ_3/n	$6.7 \cdot 10^{-13}$	$8.6 \cdot 10^{-11}$	$3.5 \cdot 10^{-6}$
4	τ_4/n	0.001	0.001	0.001
5	τ_5/n	0.001	0.003	0.007

Among the considered scenarios, Scenario 5 leads to the highest disclosure risk. However, the regression model in this scenario comes with high computational costs and the probabilities of re-identification are still far too low to obtain any useful information.

The results show that while re-identification is possible, an intruder would not gain any useful information from the purely synthetic data.

Even if they successfully re-identify a unique combination of key variables from their data, the probability that the obtained value is close to the original value is extremely low for all proposed worst-case scenarios given in the paper.

Due to our experiences and the results from the investigated scenarios, we can argue that synthetic population data generated with the methodology introduced in Alfons et al. (2011), Templ and Filzmoser (2014), Templ et al. (2017), described in this chapter, and implemented in **simPop** are confidential and can be distributed to the public.

Question 6.1 Estimate the global risk for data set `eusilc` as done above. Then use a subset of `eusilc` and compare the results on the global disclosure risk. Do smaller data sets imply higher risks?

Question 6.2 Estimate the global risk for data set `eusilc` as done above. Then use a subset of `eusilc` and compare the results on the global disclosure risk. Do smaller data sets imply higher risks?

6.4 Data Utility of Synthetic Data

There is not much to say in addition to Chap. 5 since the utility can be estimated with the same methods. However, we want to point out a few special characteristics of synthetic data that are produced with the methods proposed in this chapter.

Generally, in practical applications we observe that synthetic populations shows excellent quality in categorical variables simulated. Also the quality of continuous and semi-continuous variables and their relation to the categorical variables is usually excellent. Only in case of components, especially for sparse components the results might not always be satisfactory. No solution can be found in the literature except those implemented described in this chapter to calculate ratios between the components and the total in the original file and took this ratios to estimate the values of the components in the synthetic file.

Sometimes the quality of the relation between continuous information on primary sampling units (e.g. household income) and continuous variables on secondary sampling units (e.g. personal income) is problematic. This can have three reasons. First, missing values plays always a huge role. In general, the inclusion of missing values in the modelling process is a complex task. Second, the synthetic variables on the secondary sampling units is typically produced with another model than used for primary sampling units. This leads to discrepancies.

For such complex data, synthetic data are no replacement for traditional anonymization methods when the aim is to provide high-quality data, e.g. scientific-use files. However, since the disclosure risk is very low and since categories might be the same as for the variables of the raw data, synthetic data are an ideal candidate for teaching purposes, for remote execution purposes, for design-based simulation studies, and for producing augmented populations for microsimulation purposes.

We come back to the topic on data utility in Sect. 8.7 within a practical application.

Exercises:

Question 6.3 **When producing synthetic data sets**
Suppose you must give a scientific-use file containing detailed information on employers and employees to a group of researchers where

(a) their aim is to analyse the data, to produce detailed tables, and to run regression models to further analyse the data set and subjects of interest.
(b) their aim is to develope new methodology and compare it to existing methodology.

Would you provide them a data set that is anonymized using traditional methods or a synthetic data set?

Question 6.4 **When producing synthetic data sets**
Suppose you have to provide very detailed data for lecturing. Would you use traditional methods or methods for synthetic data simulation?

Question 6.5 **Simulate a population**
Use the data set `testdata` from **sdcMicro** and produce a synthetic population.

Question 6.6 **Check the quality of the population**
Use the synthetic population simulated in question 6.5 and generate one diagnostic plot (out of many possible ones) and compare it with the same plot for the original `testdata`.

References

Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, *22*(2), 203–217.

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., & Filzmoser, P. (2013). Robust estimation of economic indicators from survey samples based on pareto tail modelling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *62*(2), 271–286. doi:10.1111/j.1467-9876.2012.01063.x. ISSN 1467-9876.

Barrett, C.L., Eubank, S., Marathe, A., Marathe, M.V., Pan, Z., & Swarup, S. (2011). Information integration to support model-based policy informatics. *The Innovation Journal: The Public Sector Innovation Journal*, *16*(1).

Barthelemy, J., & Toint, P. L. (2013). Synthetic population generation without a sample. *Transportation Science*, *47*(2), 266–279.

Beckman, R. J., Baggerly, K. A., & McKay, M. D. (1996). Creating synthetic baseline populations. *Transportation Research Part A: Policy and Practice*, *30*(6), 415–429.

Breiman, L. (2001). Random forest. *Machine Learning*, *45*, 5–32.

Brown, L., & Harding, A. (2002). Social modelling and public policy: Application of microsimulation modelling in Australia. *Journal of Artificial Societies and Social Simulation*, *5*(4), 6.

Drechsler, J., Bender, S., & Rässler, S. (2008). Comparing fully and partially synthetic datasets for statistical disclosure control in the German IAB Establishment Panel. *Transactions on Data Privacy*, *1*(3), 105–130.

Drechsler, J. (2011). *Synthetic data sets for statistical disclosure control*. New York: Springer.

Duncan, G. T., & Lambert, D. (1986). Disclosure-limited data dissemination. *Journal of the American Statistical Association*, *81*, 10–28.

Embrechts, P., Klüppelberg, G., & Mikosch, T. (1997). *Modelling extremal events for insurance and finance*. New York: Springer.

Fienberg, S. E., Makov, U. E., & Sanil, A. P. (1997). A bayesian approach to data disclosure: optimal intruder behavior for continuous data. *Journal of Official Statistics*, *13*, 75–89.

Horvitz, D. G., & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, *47*(260), 663–685.

Hothorn, T., Hornik, K., van de Wiel, M. A., & Zeileis, A. (2006a). A lego system for conditional inference. *The American Statistician*, *60*(3), 257–263.

Hothorn, T., Hornik, K., & Zeileis, A. (2006b). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674.

Kleiber, C., & Kotz, S. (2003). *Statistical size distributions in economics and actuarial sciences*. Hoboken: Wiley.

Kraft, S. (2009). Simulation of a population for the European income and living conditions survey. Master's thesis, Department of Statistics and Probability Theory, Vienna University of Technology, Vienna, Austria.

Little, R. J. A. (1993). Statistical analysis of masked data. *Journal of Official Statistics*, *9*(2), 407–426.

Münnich, R., Schürle, J., Bihler, W., Boonstra, H.-J., Knotterus, P., Nieuwenbroek, N., et al. (2003a). Monte carlo simulation study of European surveys. In *DACSEIS Deliverables D3.1 and D3.2*, University of Tübingen.

Münnich, R., Schürle, J., Bihler, W., Boonstra, H.-J., Knotterus, P., Nieuwenbroek, N., et al. (2003b) Monte carlo simulation study of European surveys - DACSEIS deliverables. Technical report, University of Tübingen. http://www.dacseis.de.

Raghunathan, T. E., Reiter, J. P., & Rubin, D. B. (2003). Multiple imputation for statistical disclosure limitation. *Journal of Official Statistics*, *19*(1), 1–16.

Reiter, J. P. (2009). Using multiple imputation to integrate and disseminate confidential microdata. *International Statistical Review*, *77*(2), 179–195.

Rinott, Y., & Shlomo, N. (2007). A smoothing model for sample disclosure risk estimation. *JSTOR. Lecture Notes-Monograph Series*, *54*, 161–171.

Rubin, D. B. (1993). Discussion of statistical disclosure limitation. *Journal of Official Statistics*, *9*(2), 461–468.

Simonoff, J. S. (2003). *Analyzing categorical data*. New York: Springer.

Smith, D. M., Pearce, J. R., & Harland, K. (2011). Can a deterministic spatial microsimulation model provide reliable small-area estimates of health behaviours? An example of smoking prevalence in new zealand. *Health Place*, *17*(2), 618–624.

Templ, M., & Alfons, A. (2010). Disclosure risk of synthetic population data with application in the case of EU-SILC. In *Privacy in Statistical Databases.* Lecture Notes in Computer Science, pp. 174–186. Springer. ISBN 978-3-642-15837-7.

Templ, M., Meindl, B., Kowarik, A., & Dupriez, O. (2017). Simulation of synthetic complex data: The R-package simPop. *Journal of Statistical Software*, 1–38. Accepted for publication in December 2015.

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.

Tomintz, M. N., Clarke, G. P., & Rigby, J. E. (2008). The geography of smoking in leeds: Estimating individual smoking rates and the implications for the location of stop smoking services. *Area*, *40*(3), 341–353.

Weisberg, S. (2005). *Applied linear regression* (3rd ed.). Hoboken: Wiley.

Williamson, P., Mitchell, G., & McDonald, A. T. (2002). Domestic water demand forecasting: Astatic microsimulation approach. *Water and Environment Journal*, *16*(4), 243–248.

Chapter 7
Practical Guidelines

Abstract This section offers some guidelines on how to implement traditional SDC methods in practice. A rough workflow is presented and described. In addition, a brief discussion on the selection of key variables, the acceptable risk of disclosure and the choice of SDC methods should guide the user to find the best methodology.

7.1 The Workflow

Figure 7.1 presents a rough representation of a common workflow for applying SDC. Note that the figure only includes SDC methods introduced in this book.

Pre-processing steps are crucial, including the discussion on possible disclosure scenarios, the selection of direct identifiers, key variables and sensitive variables, as well as to determine an acceptable disclosures risk and levels of information loss/data utility. Several crucial points are marked in Fig. 7.1 as numbers:

1. the actual SDC process starts with deletion of direct identifiers.
2. for non-synthetic methods, the choice of key variables is of high importance.
3. for categorical key variables, before applying any SDC techniques, the disclosure risks of the original data, including record-level and global disclosure risks should be estimated. This will allow the identification of records with high disclosure risks, such as those violating k-anonymity (typically 3-anonymity) or observations with high individual risk. For continuous key variables, e.g. distance-based disclosure risk estimation should be made. Also global disclosure risk should be estimated using either a log-linear modelling approach or the sum of individual risks.
4. depending on the kind of key variables (categorical or continuous) SDC techniques can then be applied. For categorical key variables this might be local suppression to guarantee k-anonymity or any swapping technique. For continuous key variables this might be shuffling, microaggregation or adding noise or a combination of those techniques.
5. every time an SDC technique is applied, the same disclosure risk measures should also be applied and the extent of information loss should be reported. For example, how many values have been suppressed, categories combined or how the estimates on the most important indicators are different.

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_7

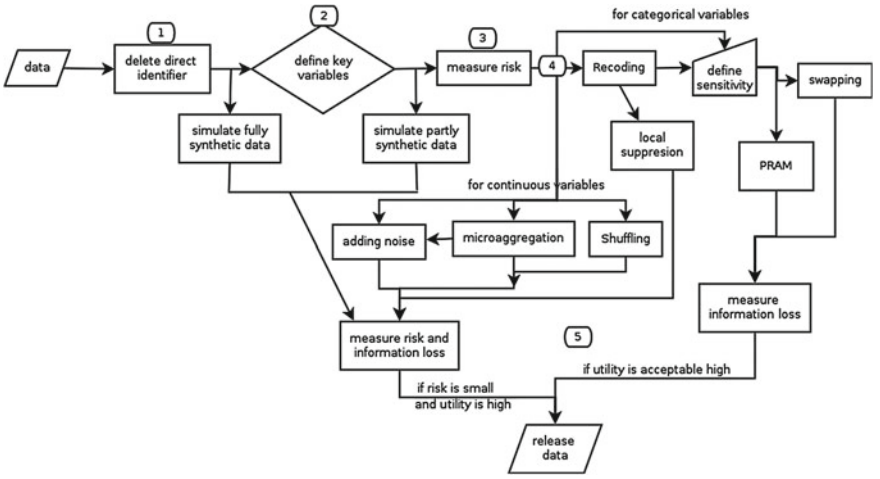


Fig. 7.1 From original data to released confidential data: simplified workflow. Originally published in Templ et al. (2015). Published with kind permission of © Matthias Templ, Alexander Kowarik and Bernhard Meindl 2015 under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium provided the original author(s) and source are credited

For continuous key variables, the disclosure risk are measured by the extent to which original and perturbed data sets can be matched, and thus the disclosure risk is by default 100% for the original data set. After applying any SDC method, disclosure risks can be measured using record linkage approaches. The disclosure risk measure indicates the percentage of the records in the perturbed data set that can be "linked" to those in the original data. The risk measure should be compared and assessed together with information loss measures. For both categorical and continuous key variables, information loss should be quantified not only by direct measures, but also by estimating indicators or models. Data are ready to be released when an acceptable level of disclosure risk has been achieved with minimal information loss. Otherwise, alternative SDC techniques should be applied and/or the same techniques should be repeated with different parameter settings.

In this chapter, we provide some practical guidelines on common questions, such as how to determine key variables and assess levels of risks, and how to select the best SDC methods for a given data set.

7.2 How to Determine the Key Variables

Most disclosure risk assessment and SDC methods rely on the selected key variables, which correspond to certain disclosure scenarios. In practice, determining key variables is a challenge, as there are no definite rules and any variable potentially

belongs to key variables, depending on the disclosure scenario. The recommended approach is to consider multiple disclosure scenarios and discuss with subject matter specialists which scenario is most likely and realistic. A common scenario is where the intruder links the released data with external data sources. Therefore, an important step is to take inventory of what other data sources are available and identify variables which could be exploited to link to the released data. In addition, sensitive variables containing confidential information should also be identified beforehand. In any case a close collaboration with the SDC expert and the subject matter specialists is necessary to select those critical variables.

7.3 The Level of Disclosure Risk Versus Information Loss

Assessment of data utility, especially for estimating indicators or models, requires knowledge about the end users of anonymized data, how they will use the released data and, as a result, what information must be preserved. If a microdata dissemination policy exists, the acceptable level of risk varies for different types of files and access conditions (Dupriez and Boyko 2010). For example, public use files should have much lower disclosure risks than licensed files, whose access is restricted to specific users subject to certain terms and conditions. Moreover, a data set containing sensitive information, such as HIV/AIDS information, might require a larger extent of perturbation, compared to that containing general, non-sensitive information.

In any case, questions on the level of disclosure risk must be answered by the management of an organisation and their lawyers and experts on national laws on SDC. For example, data of the Austrian population register might be swapped with a much lower swapping rate than in another country. The level of the swapping rate is a political and law-based decision. The user cannot be sure if an observation has been partly swapped. But the probability of being not sure (the swapping rate) is not the decision of a statistician. Another example is the EU-SILC data. Some countries in the EU sells almost unmodified EU-SILC data as scientific-use files (few recoding were done), while e.g. in France this would be unacceptable according to their lawyers.

The statisticians or SDC experts best choice is to consult the experts in laws on privacy and the management, and to discuss and fix the maximum tolerable disclosure risk with them. As soon as these numbers are fixed, the SDC experts can now modify the data to reduce the risk below the threshold.

7.4 Which SDC Methods Should Be Used

The strength and weakness of each SDC method are dependent on the structure of the data set and key variables under consideration. The recommended approach is to apply different SDC methods with varying parameter settings in an exploratory manner. Documentation of the process is thus essential to make comparisons across

methods and/or parameters and to help data producers decide on the optimal levels of information loss and disclosure risk. The following paragraphs provide general guidelines.

For categorical key variables, recoding is most commonly used. If the disclosure risks remain high after recoding, one can apply local suppression to further reduce the number of sample uniques. Recoding should be applied in such a way, so that minimal local suppression is needed afterwards. Again, this process of finding good recodings is not based on a mathematical optimization function, it is rather an explorative approach with subject matter knowledge on the data set to be anonymized. Recoding must make sense with regard to contents and subjects, and at the same time, they should considerably reduce the risk. There exist algorithms that apply a kind of "minimal" recoding, e.g. the Mondrian algorithm (LeFevre et al. 2006). However, Mondrian may not be applied since recoding should preferable never be done with an optimality criterion that is not subject matter related. For example, recoding of age into four age groups ([0–19]; [20–39]; [40–59], [60–100]) is not recommended when the most important information for socio-demographic analysis is age. It is then better to not recode age but other variables or/and to add some noise to age. Another example is the modification of regional information. When the aim of the researchers is spatial analysis, they cannot work with data of which the regional information is recoded to very broad regions. For example, for statistics on commuters, detailed information on the place of living and work is needed. Here it is better to apply (with care) swapping procedures like PRAM. Additionally, categorical key variables might be included where there is no need to have detailed information. For example, when the researcher's aim is to compare countries, a more detailed information than country level is not needed and thus information on region can be recoded to country level.

In general, if a data set has a large number of categorical key variables and/or a large number of categories for the given key variables (e.g., location variables), recoding and suppression might lead to too much information loss. In these situations, PRAM might be a more advantageous approach. PRAM can be applied with or without prior recoding. If PRAM is applied after recoding, the transition matrix should specify lower probability of swapping. In addition, for sensitive variables violating l-diversity, recoding and/or PRAM are useful methods to increase the number of distinct values of sensitive variables, for each group of observations sharing the same pattern of key variables.

In case of many key variables or in case of providing public-use-files with very low risk of disclosure, the generation of synthetic data sets is a good alternative. In addition, the generation of synthetic populations has a great number of further advantages on particular simulation tasks (see Chap. 6, Sect. 6.1).

For continuous variables, micro-aggregation is a recommended method. For more experienced users, shuffling provides promising results if one is able to find a well-fitting regression model that predicts the values of sensitive variables using other variables present in the data set (see also Muralidhar and Sarathy 2006).

For measuring the disclosure risk, in any case k-anonymity should be checked. In addition, the individual and household risks should be estimated (see Chap. 3,

Sects. 3.5 and 3.6), and further suppression and recodings should be made for individuals with high risk. Also the global risk measures (see Chap. 3, Sect. 3.7) give an impression on the global risk of a data set; they are also helpful to compare anonymized data sets, i.e. resulting data when a set of methods applied to raw survey data with another set of SDC methods applied to the same raw data set.

When producing synthetic data, it may not be necessary to look on the disclosure risk, since it should be by definition very low. However, some matching with external data or even with the raw survey data can be made to evaluate the risk of disclosure.

Exercises:

Question 7.1 **Key variables for EU-SILC**
The EU-SILC data are anonymized in Sect. 8.7 by providing synthetic data. However, let us consider the case of applying traditional (non-synthetic) methods to anonymize a country data set of EU-SILC. Which variables do you consider as key variables?

Question 7.2 **Alternatives for EU-SILC**
Considering that a scenario with many key variables would need heavy anonymizations/perturbations. What is the alternative of applying global recoding, PRAM, local suppression, microaggregation, adding noise, rank swapping and shuffling?

Question 7.3 **Key variables for the labor force survey**
The labor force survey is one key data set in official statistics. The variables described can be easily found on the Internet just by searching *lfs* and *codebook* in a search machine. Which variables would you consider as key variables?

Question 7.4 **Acceptable level of disclosure risk for socio-economic data**
Assume that you have socio-economic survey data, for example, EU-SILC. How many and which methods provide an acceptable level anonymization in terms of disclosure risk?

Question 7.5 **Acceptable level of disclosure risk for business data**
Is the situation for business survey data in general different?

Question 7.6 **Anonymization methods for EU-SILC**
Depending on your selection of key variables, which anonymization methods would you apply to EU-SILC?

References

Dupriez, O., & Boyko, E. (2010). Dissemination of microdata files. Formulating policies and procedures. IHSN Working Paper No 005. Paris: International Household Survey Network.

LeFevre, K., DeWitt, D.J., & Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *Proceedings of the 22d International Conference on Data Engineering, ICDE 2006*, Washington, DC, USA (p. 25). IEEE Computer Society. ISBN 0-7695-2570-9. http://dx.doi.org/10.1109/ICDE.2006.101.

Muralidhar, K., & Sarathy, R. (2006). Data shuffling—A new masking approach for numerical data. *Management Science*, *52*(2), 658–670.

Templ, M., Meindl, B., & Kowarik, A. (2015). Statistical disclosure control for micro-data using the R package sdcMicro. *Journal of Statistical Software*, *67*(1), 1–37.

Chapter 8
Case Studies

Abstract In this section, we show how to apply the concepts and methods introduced in the previous chapters using **sdcMicro**. Anonymized data are produced for the Family Income and Expenditure Survey (FIES), the Structural Earnings Survey (SES), International Income Distribution Database (I2D2), the Global Purchasing Power Parities and Real Expenditures data (P4), the so-called SHIP data, and the European Union Statistics on Income and Living Conditions (EU-SILC) data.

8.1 Practical Issues

The first step in creating safe microdata is of course a very accurate analysis of the raw data set to get an idea about key-characteristics and possible use of the data. Furthermore it is also necessary to take into account legal considerations. Especially it is important to define which properties are required from a microdata set that it can be considered safe. Since legal regulations vary from country to country it is not possible to define general rules. These judgments have to be made by subject matter and/or legal experts.

It is worth mentioning that the anonymization methods applied on data that are saved in flat files. In particularly, the FIES data stored in a `csv` text format, the SES data in SPSS's `sav` binary format, the I2D2 and P4 in Stata's `dta` binary format.

The general approach is to create safe microdata files according to the workflow in Sect. 7.1:

1. define a set of key variables and thus a disclosure risk scenario.
2. assess the disclosure risk of the unmodified data.
3. possibly perform recoding of variables.
4. achieve k-anonymity within the key variables by local suppression.
5. reassess disclosure risk after local suppression.
6. protect continuous variables using suitable methods.
7. remove additional information from microdata set if it is necessary.

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4_8

In Sects. 8.2 and 8.3 we will show how to transform a microdata set into a safe microdata file that could be published. It must be noted however that the purpose of this document is to create guidelines and not to actually create a safe microdata set.

8.2 Anonymization of the FIES Data

8.2.1 FIES Data Description

The *Family Income and Expenditure Survey* (FIES) from 2006 data has been provided by the National Statistics Office of the Philippines. It is a household survey to gather information on family income and expenditures and it is also been used to measure inequality. The data set consists of 38,483 observations. The survey considers the country's 17 administrative regions. It is a nationwide survey of households that is the main source of information of data on family income and expenditures.

The objectives of the survey are—among others—to gather information on family income and expenditures and related information that affects income and expenditure levels and patterns. It is also of great interest to collect information about different sources of income, levels of living and spending patterns and also about the degree of inequality among families.

For the FIES in 2006, households have been sampled according to a complex sampling design. With respect to the goal of this work which is to draft practical guidelines for creating protected microdata files it is not required to go into details about the sampling procedure. It is however important to note that sampling weights are available and should be taken into account.

The questionnaire was split into four main parts that are listed below:

- Identification and other Information
- Expenditures
- Income
- Entrepreneurial Activities

The required interviews have been conducted face to face by trained interviewers. The reporting unit was the family. The data set features a total of 38,483 units for which 721 variables have been measured.

More metadata on the FIES2006 data are available from www.census.gov.ph/nsoda, including detailed variable descriptions. In addition, by registration it is possible to get access to this data set.

8.2.2 Pre-processing Steps

The first step consists of loading the data into R. In this case the microdata are available in a `csv` format file and can easily be loaded as shown in the following code. The original data consists of hundreds of variables which we already avoided to read in, i.e. only the important variables for disclosure control are included in this data set.

First, the data set is imported to R and then the structure of variables are exploited. We also remove the first variable since it is useless.

```
fies06 <- read.csv("data/FIES06.csv")
str(fies06)

   ## 'data.frame': 38483 obs. of 12 variables:
   ## $ X : int 1 2 3 4 5 6 7 8 9 10 ...
   ## $ w_regn : int 14 14 14 14 14 14 14 14 14 14 ...
   ## $ z2011_h_sex : int 1 1 1 1 1 1 1 1 1 1 ...
   ## $ z2021_h_age : int 30 51 51 24 59 78 54 38 33 26 ...
   ## $ z2041_h_educ: int 4 4 4 2 1 1 4 5 3 1 ...
   ## $ wages : int 41000 0 95832 3120 76400 0 0 83300 74800 14240 ...
   ## $ wsag : int 0 0 0 3120 0 0 0 0 16800 9240 ...
   ## $ wsnag : int 41000 0 95832 0 76400 0 0 83300 58000 5000 ...
   ## $ pnsns : int 0 0 0 0 0 0 0 0 0 0 ...
   ## $ toinc : int 51685 172046 197190 30856 133545 72491 85662 177876
   116631 48599 ...
   ## $ rfact : num 185 185 185 185 185 ...
   ## $ z2031_h_ms : int 2 2 2 2 4 2 2 2 2 2 ...

fies06 <- fies06[, -1]
str(fies06)


   ## 'data.frame': 38483 obs. of 11 variables:
   ## $ w_regn : int 14 14 14 14 14 14 14 14 14 14 ...
   ## $ z2011_h_sex : int 1 1 1 1 1 1 1 1 1 1 ...
   ## $ z2021_h_age : int 30 51 51 24 59 78 54 38 33 26 ...
   ## $ z2041_h_educ: int 4 4 4 2 1 1 4 5 3 1 ...
   ## $ wages : int 41000 0 95832 3120 76400 0 0 83300 74800 14240 ...
   ## $ wsag : int 0 0 0 3120 0 0 0 0 16800 9240 ...
   ## $ wsnag : int 41000 0 95832 0 76400 0 0 83300 58000 5000 ...
   ## $ pnsns : int 0 0 0 0 0 0 0 0 0 0 ...
   ## $ toinc : int 51685 172046 197190 30856 133545 72491 85662 177876
   116631 48599 ...
   ## $ rfact : num 185 185 185 185 185 ...
   ## $ z2031_h_ms : int 2 2 2 2 4 2 2 2 2 2 ...
```

We observe that in the data set no direct identifiers such as names or addresses exist. Thus it is not required to remove any variables from the data set now. Looking at the data and its metadata (available in `xml`-file format) we learn that the statistical units are households and the microdata set is the result of a survey. It is therefore important to take survey weights into account if we want to asses the disclosure risk. The variable holding sampling weights is called “rfact”.

The next step consists of deciding on a set of key variables that the attacker could use to identify households. In this case we arbitrarily[1] choose the following variables to be key variables.

- **w_regn**: region with 17 characteristics
- **z2011_h_sex**: sex with 2 characteristics
- **z2021_h_age**: age with 86 characteristics
- **z2041_h_educ**: educational attainment with 21 characteristics.

8.2.3 *Frequency Counts and Disclosure Risk*

There exist two ways of applying functions in **sdcMicro**. The first one is to apply the corresponding functions (like `measure_risk()` to the data and specify needed function arguments. The second possibility is to apply the methods on a pre-defined object of class *sdcMicroObj*. We concentrate on the later possibility. The following code block shows how to create an object of class *sdcMicroObj* from the FIES 2006 data.

```
require(sdcMicro)
sdc <- createSdcObj(fies06,
                    keyVars = c('w_regn','z2011_h_sex',
                                'z2021_h_age','z2041_h_educ'),
                    numVars=c('wages','wsag','wsnag','pnsns','toinc'),
                    weightVar='rfact')
```

Hereby, the key variables have to be specified as well as the vector of sampling weights and possible stratification.

We easily then get information on frequencies, e.g. using the function `print()` that is defined for objects of class *sdcMicroObj*.

```
print(sdc)

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 4506 (11.709%)
   ##   - 3-anonymity: 8068 (20.965%)
   ##   - 5-anonymity: 13753 (35.738%)
   ##
   ## ----------------------------------------------------------------------
```

We see that a total of 4506 keys is unique in the sample ($f_k = 1$) and additionally 3562 combinations of key variables violates 3-anonymity. The corresponding percentages are also reported.

Note that the corresponding estimates for population frequencies are also stored in the object `sdc`. They are estimated with help of the sampling weights and therefore

[1] since we do not have any knowledge which information might be available to attackers in the Philippines from registers or other data sources.

not suitable for risk estimation (for details have a look at Templ 2009). Therefore a theoretical distribution for the frequency counts of the population with respect to the frequency counts in the sample is usually chosen and theoretical values of this super-population distribution are used for risk estimation (for details, see Rinott and Shlomo 2006). All the risk estimations can be accessed by `get.sdcMicroObj` (sdc, "risk") and the `print` (sdc, "risk") function gives a short summary of the global risk (obtained from individual risk). It is shown in the following code block that we can expect around 88 observations (out of 38483) to be reidentified.

```
print(sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the data:
   0
   ## Expected number of re-identifications: 91.31 (0.24 %)
```

To estimate the global risk, also log-linear models can be applied within **sdcMicro**. Note that other form of models are possible, also including interaction terms.

```
form <- ~w_regn + z2011_h_sex + z2021_h_age + z2041_h_educ +z2031_h_ms
sdc <- modRisk(sdc, formulaM = form)
slot(sdc, "risk")$model

   ## The estimated model (using method 'default') was:
   ##  ~ w_regn + z2011_h_sex + z2021_h_age + z2041_h_educ + z2031_h_ms
   ## global risk-measures:
   ##  Risk-Measure 1: 0.433 (43.300 %)
   ##  Risk-Measure 2: 0.485 (48.529 %)
```

It can be concluded that—given the disclosure scenario defined—the disclosure risk is high. Thus the data needs further protections/modifications to reduce the disclosure risk.

8.2.4 Recoding

If the disclosure risk still should be decreased, our next goal might be to achieve to reduce the disclosure risk and ensure that fewer observations violating 2 or 3-anonymity. To accomplish these goals several possibilities exist. One possibility is to recode key variables in order to reduce the number of keys. Having a look at the key variables we choose it makes sense to reduce the number of characteristics for variables age and attained education since these variable have a lot of characteristics. The following code shows how to recode variable 'z2021_h_age' into 10-year age categories and reducing information in variable 'z2041_h_educ' by combining categories.

```
sdc <- globalRecode(sdc, column="z2021_h_age",
                    breaks=seq(9,99,10), labels=1:9)
sdc <- globalRecode(sdc, column="z2041_h_educ",
                    breaks=c(-100,59,69,180), labels=c(0,6,7))
print(sdc, "kAnon")

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 84 (0.218%) | in original data: 4506 (11.709%)
   ##   - 3-anonymity: 158 (0.411%) | in original data: 8068 (20.965%)
   ##   - 5-anonymity: 334 (0.868%) | in original data: 13753 (35.738%)
   ##
   ## ----------------------------------------------------------------------
```

From the previous code, on Line 1 it is shown how to apply function `globalRecode()` to create 10-year age categories from an integer scaled variable. In this case the first age group is 10–19 years, the second group is 20–29 years. In lines 7–9 it is shown how to recode the educational status. In line 2, characteristics 6, 53 and 58 for which no formal description is given are combined with characteristic 0 (no grade completed), characteristics 60–69 which all represent some kind of bachelor degree are combined into a new category (6), and characteristics 70–79 which represent some kind of post graduation are combined into a new single category (7).

The recoding greatly reduces the number of possible combinations of the key variables. We can observe that the number of keys that are occupied by one or two households is much lower than before, the risk is reduced heavily.

```
print(sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the data:
   ##   in modified data: 0
   ##   in original data: 0
   ## Expected number of re-identifications:
   ##   in modified data: 2.75 (0.01 %)
   ##   in original data: 91.31 (0.24 %)
```

8.2.5 Local Suppression

Naturally, also 3-anonymity should be achieved. If so, we may apply local suppression. In order to gain only few suppressed values, we have already done global recoding on some of the key variables. Using **sdcMicro** k-anonymity can easily be achieved by using function `localSuppression()`. (Remember that with function `localSupp()` one can apply suppression based on individual risk.)

The following code shows how to apply local suppression. Note that using the defaults, all categorical key variables are selected and $k = 3$-anonymity is reached, whereas the importance of variables is determined automatically (but can be

overwritten). It can be seen that 49 suppression are done in age, and 10 in education.

```
sdc <- localSuppression(sdc, k = 3)   ## sensible defaults
print(sdc, "ls")

   ## Local suppression:
   ##        KeyVar | Suppressions (#) | Suppressions (%)
   ##        w_regn |               43 |            0.112
   ##   z2011_h_sex |                0 |            0.000
   ##   z2021_h_age |                3 |            0.008
   ##  z2041_h_educ |                0 |            0.000
   ## ---------------------------------------------------------------------------
```

Let's validate if 3-anonymity is reached.

```
print(sdc, "kAnon")

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 0 (0.000%) | in original data: 4506 (11.709%)
   ##   - 3-anonymity: 0 (0.000%) | in original data: 8068 (20.965%)
   ##   - 5-anonymity: 147 (0.382%) | in original data: 13753 (35.738%)
   ##
   ## ---------------------------------------------------------------------------
```

The global risk is again reduced, which can be proofed by the following output.

```
print(sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the data:
   ##   in modified data: 0
   ##   in original data: 0
   ## Expected number of re-identifications:
   ##   in modified data: 1.28 (0.00 %)
   ##   in original data: 91.31 (0.24 %)
```

8.2.6 Perturbing the Continuous Key Variables

We now continue to protect confidential numerical variables. In this microdata set, we decide that all variables containing information on income should be deemed confidential and thus be additionally protected while numerical variables containing information on expenditures are not confidential. To protect numerical confidential variables one could use the function `microaggregation()` as shown in the following code.

```
sdc <- microaggregation(sdc)
print(sdc, "numrisk")

   ## Numerical key variables: wages, wsag, wsnag, pnsns, toinc
   ##
   ## Disclosure risk (~100.00% in original data):
   ##   modified data: [0.00%; 95.37%]
   ##
   ## Current Information Loss in modified data (0.00% in original data):
   ##   IL1: 0.01
   ##   Difference of Eigenvalues: -72.770%
   ## ------------------------------------------------------------------------
```

In the first line of the previous code we already applied microaggregation using method 'mdav', which forms the proximity using a multivariate distance method. We use the default parameter `aggr = 3` so that proximities of size 3 are generated. Note that from size 3 automatically 3-anonymity is created, i.e. always three observations have the same values in the mircoaggregated variables. More protection is guaranteed if the aggregation level is increased but the data utility will be decreased at the same time.

8.2.7 PRAM

It is now of course possible to add another layer of uncertainty to the data set by stochastically changing characteristics of some (categorical) variables using post-randomization with function `pram()` as it is shown in Listing 8.1.

```
sdc <- pram(sdc, keyVar='w_regn')
head(get.sdcMicroObj(sdc, "pram"))
  transition Frequency
1     1 --> 1      1967
2    1 --> 10        17
3    1 --> 11        19
4    1 --> 12        31
5    1 --> 13        14
6    1 --> 14        11
```

Listing 8.1 Post randomization for a categorical variable and first six lines of pram summary on transitions.

In the first line of Listing 8.1 `pram` is applied to variable 'w_regn' (region). We did not changed the defaults of pram, e.g. parameter 'pd' is set to 0.8 per default, which means that on average around 80% of the variable characteristics should not change. Decreasing this parameter value more values are changed to other categories and the data utility decreases as well as the disclosure risk. Again, this parameter should be specified on legal basis, i.e. the need for changing more values than 0.15%, for example, should be argued by lawyers that can decide how many changes are sufficient so that an intruder is (enough) unsure if the value is true or not.

The transition matrix itself is internally calculated. The resulting *sdcMicroObj* object contains besides the transition matrix, the original input vector and the modified, post-randomized vector.

8.2.8 Remark

We can now state that we have created a microdata set that features 3-anonymity within the selected set of key variables that has low individual re-identification risks and that has been microaggregated in all variables that contain information on any sort of income which we defined as confidential. Additionally we have post-randomized another categorical variable. After performing these steps it would be the task of the subject matter specialists or people responsible to decide if the modified microdata set is safe to be published. If this is the case, the file could be published. If not, additional disclosure limitation techniques such as adding noise (using `addNoise()` or `shuffling()`) should be applied or the information of the microdata set could be even more reduced by additional suppression, recoding, post-randomization or any other method.

Note that the global risk (function `measure_risk()` and `LLmodGlobalRisk()`) can be applied to the data. An application of that is shown in the next example using the SES data.

8.3 Application to the Structural Earnings Statistics (SES) Survey

The Structural Earnings Survey (SES) is conducted in almost all European countries and it includes variables on earnings of employees and other variables on employees and employment level (e.g. region, size of the enterprise, economic activities of the enterprise, gender and age of the employees, …).

8.3.1 General Information About SES

The Structural Earnings Survey (SES) is conducted in almost all European countries, and the most important indicators/estimates are reported to Eurostat. Moreover, also anonymised microdata are sent from most European Union countries to Eurostat.

SES is a complex survey of Enterprises and Establishments with more than 10 employees (11600 enterprises in Austria in year 2006), NACE C-O, including a large sample of employees (Austria: 207.000). In many countries, a two-stage design is

used whereas in the first stage a stratified sample of enterprises and establishments on NACE 1-digit level, NUTS 1 and employment size range is drawn with large enterprises commonly having higher inclusion probabilities. In stage 2, systematic sampling or simple random sampling is applied in each enterprise. Often, unequal inclusion probabilities regarding employment size range categories are used. Of course, calibration is applied to represent some population characteristics corresponding to NUTS 1 and NACE 1-digit level, but also calibration is carried out for gender (amount of men and women in the population).

SES includes information from different perspectives and sources. In the Austrian case this belongs to (see also Templ and Filzmoser 2014; Templ 2015):

Information on enterprise level: Question batteries are asked to enterprises like if an enterprise is private or public or if an enterprise has a collective bargaining agreement (both binary variables). As a multinomial variable, the kind of collective agreement is included in the questionnaire.

Information on individual employment level: The following questions to employees come with the standard questionnaire: social security number, date of being employed, weekly working time, kind of work agreement, occupation, time for holidays, place of work, gross earning, earning for overtime and amount of overtime.

Information from registers: All other information may come from registers like information about age, size of enterprise, occupation, education, amount of employees, NACE and NUTS classifications.

8.3.2 Details on Some Variables

Variables on Enterprise Level

The most important variables are now briefly described:

1. `Location`: The geographical location of the statistical units is cut into three areas based on NUTS 1-digit level. The three areas are AT1 (eastern Austria), AT2 (southern Austria) and AT3 (western Austria).
2. `NACE1`: The economic activity of enterprises on NACE 2-digit level.
3. `Size`: The employment size range, split into 6 categories.
4. `payAgreement`: The form of collective pay agreement consists of seven levels.
5. `EconomicFinanc`: The form of economic and financial control has two levels: A (public control) and B (private control).

Variables on Employee-Level
Categorical Variables

The most important variables are now listed:

1. `Sex`: the gender of the sampled person.
2. `Occupation`: this variable is coded according to the International Standard Classification of Occupations, 1988 version (ISCO-88(COM))at the two-digit level.
3. `education`: six categories of the highest successfully completed level of education and training are coded according to the International Standard Classification of Education, 1997 version (ISCED 97).
4. `FullPart`: the variable `FullPart` indicates if an employee is a full-time worker or a part-time worker.
5. `contract`: the categories of the type of employment contract.

Continuous Variables on Employee Level

1. `birth`: The year of birth.
2. `Length`: The total length of service in the enterprises in the reference month is based on the number of completed years of service.
3. `ShareNormalHours`: The share of a full timer's normal hours. The hours contractually worked of a part-time employee should be expressed as a percentage of the number of normal hours worked by a full-time employee in the local unit.
4. `weeks`: Represents the number of weeks in the reference year to which the gross annual earnings relate. That is the employee's working time actually paid during the year which should correspond to the actual gross annual earnings. (2 decimal places).
5. `hoursPaid`: The number of hours paid in the reference month which means these hours actually paid including all normal and overtime hours worked and remunerated by the employee during the month.
6. `overtimeHours`: The variable `overtimeHours` contains the number of overtime hours paid in the reference month. Overtime hours are those worked in addition to those of the normal working month.
7. `holiday`: The annual days of holiday leave (in full days).
8. `earnings`: Let `earnings` be gross annual earnings in the reference year. The actual gross earnings for the calendar year are supplied and not the gross annual salary featured in the contract.
9. `notPaid`: Examples of annual bonuses and allowances are Christmas and holiday bonuses, 13th and 14th month payments and productivity bonuses, hence any periodic, irregular and exceptional bonuses and other payments that do not feature every pay period. Besides the main difference between annual earnings and monthly earnings is the inclusion of payments that do not regularly occur in each pay period.
10. `earningsMonth`: The gross earnings in the reference month covers renumeration in cash paid during the reference month before any tax deductions and

social security deductions and social security contributions payable by wage earners and retained by the employer.

11. `earningsOvertime`: It is also necessary to refer to earnings related to overtime. The amount of overtime earnings paid for overtime hours is required.
12. `paymentsShiftWork`: These special payments for shift work are premium payments during the reference month for short work, night work or weekend work where they are not treated as overtime.

8.3.3 Applications and Statistics Based on SES

Every 4 years the standard publication from national statistical offices is disseminated after the survey is conducted. In addition, special publications about low incomes, non-common occupation employment and gender-specific reports are published by some member states (see, e.g., Geissberger and Knittler 2010; Geissberger 2010). Many other national publications from statistical agencies or researchers are available in almost every country (for some summaries about publications until 1999, see Belfield 1999; Nolan and Russel 2001; Dupray et al. 1999; Frick and Winkelmann 1999; Dell'Aringa et al. 2000).

It is interesting to note that anonymized SES 2002 and 2006 data (Eurostat 2014b) from 23 countries can be accessed for research purposes (by means of research contracts) through the safe centre or anonymized CD-ROM at the premises of Eurostat.[2] The output of the users are checked by Eurostat on confidentiality and quality (Eurostat 2014a).

SES Microdata from Czech Republic, Hungary, Ireland, Italy, Latvia, Lithuania, Netherlands, Norway, Portugal, Slovakia and Spain can also be analysed via the Piep Lissy remote access system. The user can run `Stata` code on the Piep-Lissy server, whereas some commands (12 commands in summary) are blocked by the system to prevent listing of individuals. This is, of course not enough to prevent re-identification of individuals (see, e.g., Templ 2011a). The Lissy servers has been intensively used within the EU project on *Linked Employer-Employee Data* (LEED) that studied the potential of linked employer-employee and panel data sets for analysis of European labor market policy. Moreover, this data set was used within the dynamic wage network that was funded by the European Central Bank.

Generally such linked employer-employee data are used to identify the determinants/differentials of earnings but also some indicators are directly derived from the hourly earnings like the gender pay gap or the Gini coefficient (Gini 1912). The most classical example is the income inequality between genders as discussed in Groshen (1991), for example.

A correct identification of factors influencing the earnings could lead to relevant evidence-based policy decisions. The research studies are usually focused on examining the determinants of disparities in earnings. Earnings comparisons among different

[2] http://epp.eurostat.ec.europa.eu/portal/page/portal/microdata/ses.

industries or regions are frequently performed (see, e.g., Stephan and Gerlach 2005; Caju et al. 2009a, b, 2010; Messina et al. 2010; Dybczak and Galuscak 2010; Simón 2010; Pointner and Stiglbauer 2010). Sometimes the socio-educational factors are investigated as possible explanatory variables of income, for example in Bowles et al. (2001). The overview of the analyses performed using SES data highlighted that, generally, the hourly log-earnings are modeled (see also Templ 2015). The explanatory variables correspond to the employer activity (related to the enterprise), his/her experience (education, length of stay in service, qualification, etc.) and working hours. It was also observed that linear models are extensively used. Anova analysis, linear mixed-effects models and multi-level models are other examples of statistical tools that have been applied. However, a lot of similar models are applied in literature to model the log hourly earnings.

For a detailed overview on the usage of the SES data, have a look at Templ (2011a, 2015).

8.3.4 The Synthetic SES Data

In this book we use a synthetic close-to-reality SES data set that has been simulated by Templ and Filzmoser (2014). This ensures that the readers can also perform the anonymization, which is hardly be possible when using original confidential SES data. In Templ and Filzmoser (2014) it is shown that these synthetic data fulfills the following properties:

- Actual sizes of regions and strata have to be reflected.
- Marginal distributions and interactions between variables are realistic and very close to the original ones. The distribution and conditional distributions of variables are realistic.
- Heterogeneities between subgroups, especially regional aspects, are present.
- All important estimates provide high quality, i.e. are very close to the original ones.
- Mosaicplots, distribution plots and many other exploratory methods applied to the synthetic data and original data result in very similar results.

The synthetic data set was simulated with the help of the R package **simPop** (Templ et al. 2017)

8.3.5 Key Variables for Re-identification

Again no direct identifiers are included in the data. The identification of an enterprise may leads to information about their employees.

For the categorical key variables at employment level the following variables are selected (see also Ichim and Franconi 2007):

- **size**: size of the enterprise
- **age**: age in years
- **location**: geographic location with 3 categories
- **economic activity**: the economic branch of the corresponding work centre given as NACE Rev. 2—Statistical classification of economic activities

As continuous key variables at employment level the following variables are selected:

- **earnings**: gross earnings.
- **hourly earnings**: generated from earnings and hours paid.

8.3.6 Pre-processing Steps

The synthetic SES data set contains already some recoding. The variable that contains information on gender has been recoded, the earnings per hour have been constructed from earnings and hours paid, and the variable 'age' has to be calculated from the year of birth. In the following the synthetic SES data set is loaded and the column names are reported.

```
library("laeken")
data(ses)
colnames(ses)

   ##  [1] "location"          "NACE1"
   ##  [3] "size"              "economicFinanc"
   ##  [5] "payAgreement"      "IDunit"
   ##  [7] "sex"               "age"
   ##  [9] "education"         "occupation"
   ## [11] "contract"          "fullPart"
   ## [13] "lengthService"     "weeks"
   ## [15] "hoursPaid"         "overtimeHours"
   ## [17] "shareNormalHours"  "holiday"
   ## [19] "notPaid"           "earningsOvertime"
   ## [21] "paymentsShiftWork" "earningsMonth"
   ## [23] "earnings"          "earningsHour"
   ## [25] "weightsEmployers"  "weightsEmployees"
   ## [27] "weights"

## size of the synthetic SES data:
dim(ses)

   ## [1] 15691    27
```

Not that the original SES data has about 200.000 observations, while the synthetic data consists of only 15691.

In the synthetic SES, the NACE classification have been changed from 2-digit codes to 1-digit codes, whereas the aggregation of the classifications are based on expert knowledge, i.e. those categories are combined where the economic branches are similar.

8.3.7 Risk Estimation

In the following code block it is shown how to set up the disclosure scenario for the SES data. First an object of class *sdcMicroObj* using `createSdcObj()` is created that includes the information on the disclosure scenario. In the resulting object, the disclosure risk of our data corresponding to the key variables is already available.

```
sdc <- createSdcObj(ses,
                    keyVars=c('size', 'age', 'location', 'occupation'),
                    numVars=c('earningsHour', 'earnings'),
                    weightVar='weights')
print(sdc, "kAnon")

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 243 (1.549%)
   ##   - 3-anonymity: 509 (3.244%)
   ##   - 5-anonymity: 1055 (6.724%)
   ##
   ## ---------------------------------------------------------------------------
```

From this output it is easy to see that 243 observations are unique in their categorical key variables (`fk = 1`. However, because of the size of the data this number is not that large as it looks at the first view, about 1.55% of the observations are unique. Nevertheless, a certain number of observations may have a considerable higher risk, about 300 individuals can be re-identified.

```
print(sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations
   ## with higher risk than the main part of the data: 547
   ## Expected number of re-identifications: 298.49 (1.90 %)
```

8.3.7.1 Recoding and Local Suppression

For the original SES data it is necessary to recode some categories of the key variables to receive a lower number of uniqueness.

The following was already done to the original SES, see also the comments in Sect. 8.3.6. It is not needed to do this on the synthetic SES, i.e. the following code is just shown for readers knowing the original SES data.

```
## recode economic activity
library(stringr)
a <- as.character(x$economicActivity)
ecoANew <- rep('Q-ExtraTerr', nrow(x))
ecoANew[a %in% str_c('R', 10:14)] <- 'C-Mining'
ecoANew[a %in% str_c('R', 15:37)] <- 'D-Manufacturing'
ecoANew[a %in% str_c('R', 38:44)] <- 'E-Electricity'
ecoANew[a %in% str_c('R', 45:49)] <- 'F-Construction'
ecoANew[a %in% str_c('R', 50:54)] <- 'G-Trade'
ecoANew[a %in% str_c('R', 55:59)] <- 'H-Hotels'
ecoANew[a %in% str_c('R', 60:64)] <- 'I-Transport'
ecoANew[a %in% str_c('R', 65:69)] <- 'J-FinancialIntermediation'
ecoANew[a %in% str_c('R', 70:74)] <- 'K-RealEstate'
ecoANew[a %in% str_c('R', 75:79)] <- 'L-Public'
ecoANew[a %in% str_c('R', 80:84)] <- 'M-Education'
ecoANew[a %in% str_c('R', 85:89)] <- 'N-Health'
ecoANew[a %in% str_c('R', 90:94)] <- 'O-Other'
ecoANew[a %in% str_c('R', 95:98)] <- 'P-Households'
sdc@manipKeyVars[,"economicActivity"] <- ecoANew; rm(ecoANew,a)

## recode size classes:
levels(sdc@manipKeyVars[,"Size"]) <- list(E10_49=c('E10_49'),
            E50_249='E50_49',
            E250plus=c('E250_499','E500_999','E1000'))

## recode age
sdc <- globalRecode(sdc, column="age",
                    breaks=c(0,19,29,39,49,59,120))
```

In general there are at least four possibilities to achieve k-anonymity. The first possibility is to randomize the values of a categorical variable with the help of function `pram()`, as shown in Sect. 4.2.3. An alternative way could be to delete some values randomly and impute those values in a proceeding step.

Another possibility is to apply further recodings, for example, to allow fewer categories for the economic activity. Another possibility is to apply local suppression.

```
sdc <- localSuppression(sdc, k = 3)
```

This results in 3-anonymity, and also the risk is reduced considerably.

```
print(sdc, "kAnon")

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 0 (0.000%) | in original data: 243 (1.549%)
   ##   - 3-anonymity: 0 (0.000%) | in original data: 509 (3.244%)
   ##   - 5-anonymity: 394 (2.511%) | in original data: 1055 (6.724%)
   ##
   ## ---------------------------------------------------------------------

print(sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the data:
   ##   in modified data: 150
   ##   in original data: 547
   ## Expected number of re-identifications:
   ##   in modified data: 165.29 (1.05 %)
   ##   in original data: 298.49 (1.90 %)
```

8.3.8 *Perturbing the Continuous Scaled Variables*

A great number of methods are available to perturb continuously scaled (key) variables.

In the following code it is shown how to apply microaggregation as well as adding (correlated) stochastic noise to continuously scaled variables. In this example the *mdav* method for microaggregation is used. The parameter `aggr` determines how many observations are always considered together when performing the aggregation.

```
## perform microaggregation
sdc <- microaggregation(sdc, aggr = 5)
print(sdc, "numrisk")

   ## Numerical key variables: earningsHour, earnings
   ##
   ## Disclosure risk (~100.00% in original data):
   ##   modified data: [0.00%; 97.02%]
   ##
   ## Current Information Loss in modified data (0.00% in original data):
   ##   IL1: 0.03
   ##   Difference of Eigenvalues: 1.520%
   ## ---------------------------------------------------------------------
```

Microaggregation has always a level of protection since any observation has at least $k = 5$ times the same values in their continuous key variables. However, if this is considered to be not enough, additionally adding noise is an option.

```
## add correlated noise
sdc <- addNoise(sdc, method='correlated2', noise = 20)
print(sdc, "numrisk")

   ## Numerical key variables: earningsHour, earnings
   ##
   ## Disclosure risk (~100.00% in original data):
   ##   modified data: [0.00%; 59.91%]
   ##
   ## Current Information Loss in modified data (0.00% in original data):
   ##   IL1: 0.12
   ##   Difference of Eigenvalues: 0.450%
   ## ------------------------------------------------------------------------
```

In addition the function `addNoise()` was applied and correlated noise added to the continuous key variables. In this case it is required to set parameter `method = 'correlated'` when calling the function. We note that quite a few different methods for noise-addition—even methods that takes the structure of outlying observations into account—can be selected in function `addNoise()`.

8.3.9 Measuring the Data Utility

Anonymized data are often evaluated using very generally defined utility measures such as differences in means or covariances. However, we already mentioned there that the evaluation of the data utility of anonymized data should better be based on those (benchmarking) statistics/indicators that are most important to estimate from the data.

In the following, the most important indicators are described in full detail, and they are evaluated on the original and anonymized data. First, the (unadjusted) gender wage gap is introduced since it is one of the most important indicator obtained from SES data, before the GINI coefficient is described.

The GINI coefficient is chosen because it is extremely sensitive to changes in the upper and lower tail of the distribution. So, if this estimator is not affected from anonymization methods, we can be confident that the data have high data utility since it is most difficult to preserve the structure of the data in the upper tail of the distribution.

Lastly, a model-based estimation on microdata level is described which is representative for all model-based estimations. Note that this choice of indicators and models is subjective, but it can be expected that differences in estimations from anonymized and original data according to that models deduce differences in similar models as well.

However, we conclude that the chosen indicators and models are representative for many other indicators and models estimated from this data set. Especially, because most of the chosen indicators are very sensitive to differences in the lower and upper tail of the distribution. To evaluate the effect of anonymization on models, the chosen model should reflect these effects, representative for (almost) any other models used in statistical agencies, at Eurostat or research institutions.

For FIES and SES, this includes that famous indicators like the GINI coefficient, but also model-based estimations that are typically applied on the data should be as precise as possible. In addition, we take a look if the variances of the estimates are preserved.

We now continue to describe how one can achieve the goal of evaluating the quality of the results. It should also be noted that it may be necessary to perform the quality assessment multiple times to be able to compare different anonymization methods. This allows the SDC specialist to decide on an anonymization technique that not only leads to protected micro data but also to data that still feature high data quality.

Finally we have chosen the following utility measures:

- Differences in the estimation of the Gender Pay Gap (GPG) and the GINI coefficient from the original and perturbed data. This can also be evaluated for h domains by

$$ARB = \frac{|\frac{1}{h}\sum_{i=1}^{h}(\hat{\theta}_i - \theta_i)|}{\theta_i} \quad . \tag{8.1}$$

- Additionally, one model is estimated.
- Moreover, the variances are estimated and the overlap of the confidence intervals of the perturbed and original data can as well be evaluated and reported in percentages (of overlap).

8.3.9.1 Comparison Regarding the GINI Coefficient

The Gini coefficient (Gini 1912) and the Quintile Share Ratio (QSR) are well known measures of inequality of a distribution and they are widely applied in many fields of research. They are often used to measure inequality of income or earnings as an economic indicator. For the SES data, the GINI and the QSR may be estimated for each enterprise, economic branch or for each country. The GINI and especially the QSR are sensitive to changes in values in the upper and lower tail of the distribution. Since large earnings have to be perturbed in a greater extent in the anonymization process than the main bulk of the data, these indicators are ideally suited to evaluate protection methods on continuous variables.

The Gini coefficient according to EU-SILC (2004, 2009) is estimated by

$$\widehat{Gini} := 100 \left[\frac{2 \sum_{i=1}^{n} \left(w_i x_i \sum_{j=1}^{i} w_j \right) - \sum_{i=1}^{n} {w_i}^2 x_i}{\left(\sum_{i=1}^{n} w_i \right) \sum_{i=1}^{n} (w_i x_i)} - 1 \right]. \tag{8.2}$$

The Gini coefficient is closely related to the Lorenz curve (Lorenz 1905), which plots the cumulative proportion of the total income against the corresponding proportion of the population.

The Gini coefficient is typically—among other domains—estimated with breakdown by age and gender or age, gender and region.

It is also important to note that also the variance components of these estimations are important to estimate since they reflect the statistical uncertainty.

As a general procedure to estimate variances, we implemented a calibrated bootstrap to estimate the variances (Templ and Alfons 2011) for the gender pay gap but also for all other indicators defined in this contribution. The calibrated bootstrap is applied internally by calling function `variance()` in package **laeken** (Templ et al. 2011b). Note that a calibrated bootstrap is preferable over all other resampling methods (Bruch et al. 2011; Templ and Alfons 2011).

Let $\boldsymbol{X}$ denote a survey sample with n observations and p variables. Then the *calibrated bootstrap algorithm* for estimating the variance and confidence interval of an indicator can be summarized as follows:

1. Draw R independent bootstrap samples $\boldsymbol{X}_1^*, \ldots, \boldsymbol{X}_R^*$ from $\boldsymbol{X}$.
2. Calibrate the sample weights for each bootstrap sample $\boldsymbol{X}_r^*, r = 1, \ldots, R$. Generalized raking procedures are used for calibration purposes: either a multiplicative method known as *raking*, an additive method or a logit method (see Deville and Särndal 1992; Deville et al. 1993) may be applied.
3. Compute the bootstrap replicate estimates $\hat{\theta}_r^* := \hat{\theta}(\boldsymbol{X}_r^*)$ for each bootstrap sample $\boldsymbol{X}_r^*, r = 1, \ldots, R$, where $\hat{\theta}$ denotes an estimator for a certain indicator of interest. Of course the sample weights always need to be considered for the computation of the bootstrap replicate estimates.
4. Estimate the variance $V(\hat{\theta})$ by the variance of the R bootstrap replicate estimates:

$$\hat{V}(\hat{\theta}) := \frac{1}{R-1} \sum_{r=1}^{R} \left(\hat{\theta}_r^* - \frac{1}{R} \sum_{s=1}^{R} \hat{\theta}_s^* \right)^2. \tag{8.3}$$

5. Estimate the confidence interval at confidence level $1 - \alpha$ by the percentile method: $\left[\hat{\theta}^*_{((R+1)\frac{\alpha}{2})}, \hat{\theta}^*_{((R+1)(1-\frac{\alpha}{2}))} \right]$, as suggested by Efron and Tibshirani (1993). $\hat{\theta}^*_{(1)} \leq \ldots \leq \hat{\theta}^*_{(R)}$ denote the order statistics of the bootstrap replicate estimates.

The Gini coefficient and its variance are estimated from both the original and the anonymized microdata. The confidence intervals are very similar and almost completely overlap. This is also true when displaying the confidence intervals by stratum (this is saved in object `v1` and `v1a`).

```
## gini from original data
g1 <- gini(inc="earningsHour",
           weigths="weights",
           breakdown="education",
           data=ses)
g1
```

```
## Value:
## [1] 28.54445
##
## Value by domain:
##         stratum    value
## 1 ISCED 0 and 1 28.32054
## 2       ISCED 2 30.07743
## 3 ISCED 3 and 4 25.64211
## 4      ISCED 5A 26.73128
## 5      ISCED 5B 23.61221
```

```
## gini from perturbed data
sesAnon <- extractManipData(sdc)
g1a <- gini(inc="earningsHour",
            weigths="weights",
            breakdown="education",
            data=sesAnon)
g1a
```

```
## Value:
## [1] 28.61509
##
## Value by domain:
##         stratum    value
## 1 ISCED 0 and 1 28.56205
## 2       ISCED 2 30.25512
## 3 ISCED 3 and 4 25.82539
## 4      ISCED 5A 26.18269
## 5      ISCED 5B 23.74283
```

We can observe that there are only larger differences the subset where ISCED is 0 and 1. However, in the following it is observable that in all cases, the standard error is much higher than the differences between the original data and the anonymized data. The confidence intervals for ISCED educational group 0 and 1 are large and thus it is not surprising that the anonymized data gives a slightly different Gini coefficient for this group.

```
## corresponding variances
v1 <- variance("earningsHour",
               weights="weights",
               data=ses,
                           indicator=g1,
                           X=calibVars(ses$location),
                           breakdown="education",
                           seed=123)
v1a <- variance("earningsHour",
                weights="weights",
                data=sesAnon,
                            indicator=g1a,
                            X=calibVars(sesAnon$location),
                            breakdown="education",
                            seed=123)
## extract the confidence intervals:
v1$ci

   ##    lower    upper
   ## 29.77703 31.44414

v1a$ci

   ##    lower    upper
   ## 29.85012 31.46251

## and for each breakdown by educational group
v1$ciByStratum

   ##         stratum    lower    upper
   ## 1 ISCED 0 and 1 21.81343 52.43583
   ## 2       ISCED 2 30.30807 33.95821
   ## 3 ISCED 3 and 4 26.22163 28.02518
   ## 4      ISCED 5A 23.95529 29.93168
   ## 5      ISCED 5B 22.00196 28.67456

v1a$ciByStratum

   ##         stratum    lower    upper
   ## 1 ISCED 0 and 1 19.96480 47.83282
   ## 2       ISCED 2 30.33500 34.00299
   ## 3 ISCED 3 and 4 26.37125 28.21482
   ## 4      ISCED 5A 24.02648 28.20864
   ## 5      ISCED 5B 22.03509 28.55613
```

8.3.9.2 The Gender Wage/Pay Gap

Probably the most important indicator derived from the SES data is the *gender pay gap*/*gender wage gap*.

The calculation of the gender pay gap is based on each person's hourly earnings. The hourly earnings equals to the gross monthly job earnings divided by the number of hours usually worked per week in job during 4.33 weeks, see (EU-SILC 2009; Beblot et al. 2003).

Definition Gender Pay Gap:

The gender pay gap in unadjusted form is defined on population level as the difference between average gross earnings of male paid employees and of female paid employees divided by the earnings of male paid employees (EU-SILC 2009).

Estimation of the Gender Pay Gap:

Since the gender wage gap is usually estimated by survey information, the estimation has to consider sampling weights in order to ensure sample representativity.

For the following definitions, let $\boldsymbol{x} := (x_1, \ldots, x_n)'$ be the hourly earnings with $x_1 \leq \ldots \leq x_n$ and let $\boldsymbol{w} := (w_i, \ldots, w_n)'$ be the corresponding personal sample weights, where n denotes the number of observations.

Let

$$J^{(M)} := \{j \in \{1, \ldots, n\} \mid \text{worked as least 1 hour per week} \wedge (16 \leq \text{age} \leq 65) \wedge \text{person is male}\} \quad ,$$

and $J^{(F)}$ those index set which differs from $J^{(M)}$ in the fact that it includes all females instead of males.

With these index sets the gender pay gap in unadjusted form is estimated by

$$GPG_{(mean)} = \frac{\frac{\sum_{i \in J^{(M)}} w_i x_i}{\sum_{i \in J^{(M)}} w_i} - \frac{\sum_{i \in J^{(F)}} w_i x_i}{\sum_{i \in J^{(F)}} w_i}}{\frac{\sum_{i \in J^{(M)}} w_i x_i}{\sum_{i \in J^{(M)}} w_i}} \quad . \tag{8.4}$$

The gender pay gap is usually estimated at domain level like economic branch, education and age groups (Geissberger 2009).

```
## gini from original data
gpg1 <- gpg(inc="earningsHour",
            weigths="weights",
            breakdown="education",
            gender = "sex",
            data=ses)
gpg1
```

```
## Value:
## [1] 0.2517759
##
## Value by domain:
##         stratum      value
## 1 ISCED 0 and 1 0.02347578
## 2       ISCED 2 0.21265286
## 3 ISCED 3 and 4 0.22974069
## 4      ISCED 5A 0.23323499
## 5      ISCED 5B 0.18445275
```

```
## gini from perturbed data
gpg1a <- gpg(inc="earningsHour",
             weigths="weights",
             breakdown="education",
             gender = "sex",
             data=sesAnon)
gpg1a

   ## Value:
   ## [1] 0.2502994
   ##
   ## Value by domain:
   ##         stratum      value
   ## 1 ISCED 0 and 1 0.06304426
   ## 2       ISCED 2 0.21061860
   ## 3 ISCED 3 and 4 0.23000629
   ## 4      ISCED 5A 0.22003603
   ## 5      ISCED 5B 0.18317083
```

Again only larger differences can be found in the educational group of ISCED 0 and 1 group. This is again not surprising, since this group has only few observations included.

```
table(ses$education)

   ##
   ## ISCED 0 and 1       ISCED 2 ISCED 3 and 4      ISCED 5A
   ##            42          3009          10997           967
   ##      ISCED 5B
   ##           676
```

Again we could also look at the confidence intervals (using function `variance`), which is skipped here.

8.3.9.3 Differences in Model Estimates

But also the differences of the regression coefficients are of interest.

Respectively for all model-based estimations at employment level we choose a similar model as described in Marsden (2010) applied within the PiEP Lissy project and which is also used in Dybczak and Galuscak (2010). They fit OLS regression models where they modeled the gross hourly earnings of workers in enterprises using age, age^2, sex, education and occupation as predictors. Since the output is very long, we reduced this model just for this reason.

Similar models are also fitted within the *wage dynamics network* of the European Central Bank (Caju et al. 2010; Pointner and Stiglbauer 2010) and within the *EU Linked Employer-Employee Project* (see, e.g., Simón 2010).

In the following the log hourly earnings are modeled by a set of predictors. Note, that no exhaustive model search are made, and no interaction terms were included. However, the following reports how different model estimates are between the original data and the modified data. First, the results for the original data are reported.

```
summary(lm(log(earningsHour) ~ location + size + sex + age + education,
    data = ses))

##
## Call:
## lm(formula = log(earningsHour) ~ location + size + sex + age +
##     education, data = ses)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9784 -0.2242  0.0318  0.2924  2.5481
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              6.431e-01  1.078e-01   5.967 2.46e-09
## locationAT2             -8.154e-02  1.225e-02  -6.654 2.94e-11
## locationAT3             -3.272e-02  9.960e-03  -3.285 1.02e-03
## sizeE10_49              -1.956e-01  1.398e-02 -13.998  < 2e-16
## sizeE250_499             2.206e-02  1.602e-02   1.377 1.68e-01
## sizeE500_999             2.019e-04  1.460e-02   0.014 9.89e-01
## sizeE50_249             -5.179e-02  1.385e-02  -3.738 1.86e-04
## sexmale                  2.561e-01  9.037e-03  28.339  < 2e-16
## age(15,29]               7.221e-01  6.602e-02  10.938  < 2e-16
## age(29,39]               1.040e+00  6.622e-02  15.710  < 2e-16
## age(39,49]               1.144e+00  6.610e-02  17.309  < 2e-16
## age(49,59]               1.214e+00  6.648e-02  18.259  < 2e-16
## age(59,120]              1.104e+00  7.589e-02  14.552  < 2e-16
## educationISCED 2         3.563e-01  8.557e-02   4.163 3.15e-05
## educationISCED 3 and 4   7.393e-01  8.516e-02   8.681  < 2e-16
## educationISCED 5A        1.146e+00  8.684e-02  13.195  < 2e-16
## educationISCED 5B        9.197e-01  8.762e-02  10.496  < 2e-16
##
## (Intercept)             ***
## locationAT2             ***
## locationAT3             **
## sizeE10_49              ***
## sizeE250_499
## sizeE500_999
## sizeE50_249             ***
## sexmale                 ***
## age(15,29]              ***
## age(29,39]              ***
## age(39,49]              ***
## age(49,59]              ***
## age(59,120]             ***
## educationISCED 2        ***
## educationISCED 3 and 4  ***
## educationISCED 5A       ***
## educationISCED 5B       ***
## ---
## Signif. codes:  0 '***' 1e-03 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5503 on 15674 degrees of freedom
## Multiple R-squared:  0.2804, Adjusted R-squared:  0.2797
## F-statistic: 381.8 on 16 and 15674 DF,  p-value: < 2.2e-16
```

In the following the model is applied on the anonymized data set.

```
summary(lm(log(earningsHour) ~ location + size + sex + age + education,
           data = sesAnon))
```

```
##
## Call:
## lm(formula = log(earningsHour) ~ location + size + sex + age +
##     education, data = sesAnon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8259 -0.2266  0.0238  0.2808  2.4392
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.737627   0.104833   7.036 2.06e-12
## locationAT2              -0.070983   0.011458  -6.195 5.98e-10
## locationAT3              -0.025082   0.009308  -2.695 7.05e-03
## sizeE10_49               -0.184085   0.013083 -14.071  < 2e-16
## sizeE250_499              0.022159   0.014994   1.478 1.39e-01
## sizeE500_999              0.004930   0.013653   0.361 7.18e-01
## sizeE50_249              -0.047462   0.012957  -3.663 2.50e-04
## sexmale                   0.251458   0.008448  29.764  < 2e-16
## age(15,29]                0.739209   0.067322  10.980  < 2e-16
## age(29,39]                1.041110   0.067487  15.427  < 2e-16
## age(39,49]                1.135229   0.067385  16.847  < 2e-16
## age(49,59]                1.201471   0.067714  17.743  < 2e-16
## age(59,120]               1.101253   0.075925  14.505  < 2e-16
## educationISCED 2          0.293266   0.080757   3.631 2.83e-04
## educationISCED 3 and 4    0.647399   0.080376   8.055 8.55e-16
## educationISCED 5A         1.052662   0.081926  12.849  < 2e-16
## educationISCED 5B         0.828768   0.082642  10.028  < 2e-16
##
## (Intercept)              ***
## locationAT2              ***
## locationAT3              **
## sizeE10_49               ***
## sizeE250_499
## sizeE500_999
## sizeE50_249              ***
## sexmale                  ***
## age(15,29]               ***
## age(29,39]               ***
## age(39,49]               ***
## age(49,59]               ***
## age(59,120]              ***
## educationISCED 2         ***
## educationISCED 3 and 4   ***
## educationISCED 5A        ***
## educationISCED 5B        ***
## ---
## Signif. codes:  0 '***' 1e-03 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5132 on 15602 degrees of freedom
##   (72 observations deleted due to missingness)
## Multiple R-squared:  0.286,  Adjusted R-squared:  0.2853
## F-statistic: 390.6 on 16 and 15602 DF,  p-value: < 2.2e-16
```

Table 8.1 Number of observations and number of variables for the investigated I2D2 data sets for countries Bangladesh, India, Senegal, Thailand, Vietnam and South Africa

	BGD05	IND99	SEN01	THA09	VNM06	ZAF07
obs	48969	591389	58637	139590	39071	105893
vars	41	41	31	37	44	37

We can observe that the estimated coefficients do not vary much between the two models. The standard error is mostly higher than the difference of these models.

8.4 I2D2

This case study shows the anonymization of data sets from the International Income Distribution Database. A disclosure scenario is defined and the disclosure risk is reported before and after anonymization of the data. For the anonymization, recoding, local suppression and microaggregation is applied. Results for various countries like Bangladesh, India, Indonesia, Thailand, Vietnam and South Africa are presented.

8.4.1 About I2D2 Data

The International Income Distribution Database (I2D2) provides a basis for the estimation of disaggregated labor market indicators. The data set includes demographic variables, information on education, labor force and welfare. The database is conducted at a large set of surveys from developing countries, presently includes more than 200 individual data sets from 1984 onwards.

For the case study we investigated data sets from six countries. The number of observations and the amount of variables are listed in Table 8.10. For this section, we exemplary use the data from Bangladesh (Table 8.1).

Due to World Bank's contract and bilateral agreements, national statistical offices have authorized the use of these data sets for this case study.

8.4.2 Disclosure Scenario/Key Variables

It is necessary to be very carefully while determining a disclosure scenario. In this case we have selected five categorical variables and one continuous variable for anonymization purposes. These variables are visible in Table 8.7. Additionally, the variables holding the sampling weights and the household IDs (hhID) are printed. Moreover, the number of strata given by cross tabulation of the chosen five key

Table 8.2 Information on categorical and numerical key variables, sampling weights, household IDs, combinations of categories, and information on strata sizes

	BGD05	IND99	THA09	VNM06	ZAF07
cat1	gender	gender	gender	gender	gender
cat2	age	age	age	age	age
cat3	marital	marital	marital	marital	marital
cat4	empstat	empstat	empstat	empstat	empstat
cat5	reg01	reg02	reg02	reg01	reg02
num	wage	wage	wage	wage	wage
weight	wgt	wgt	wgt	wgt	wgt
hhID	idh	idh	idh	idh	idh
comb	34650	161600	376200	23760	58320
avg.size	1.41	3.66	0.16	5.88	0.67

variables as well as the average strata size (arithmetic mean of the number of observations in the strata) is given. The average strata size is very low for Bangladesh and Mexico 2008 (Table 8.2).

8.4.3 Anonymization of One Example Country

In the following we exemplify the required code to anonymize the I2D2 data from Bangladesh in 2005 using the R-package `sdcMicro` (Templ et al. 2015). In this case, the STATA data set is imported into R and an *sdcMicroObj* is created by specifying important variables for statistical disclosure control. This is done in the following code.

```
require("foreign")
bgd05 <- read.dta("data/BGD_2005_I2D2.dta")
colnames(bgd05)
```

```
##  [1] "ccode"        "year"         "idh"          "idp"
##  [5] "wgt"          "strata"       "psu"          "urb"
##  [9] "reg01"        "ownhouse"     "electricity"  "toilet"
## [13] "cellphone"    "computer"     "hhsize"       "head"
## [17] "gender"       "age"          "soc"          "marital"
## [21] "ed_mod_age"   "everattend"   "atschool"     "literacy"
## [25] "educy"        "edulevel1"    "edulevel2"    "lb_mod_age"
## [29] "lstatus"      "empstat"      "ocusec"       "nlfreason"
## [33] "industry"     "occup"        "whours"       "wage"
## [37] "unitwage"     "pci"          "pci_d"        "pcc"
## [41] "pcc_d"
```

```
dim(bgd05)
```

```
## [1] 48969    41
```

```
## create sdcMicro object
bgd05sdc <- createSdcObj(
  dat = bgd05,
  keyVars = c("gender", "age", "marital", "empstat", "reg01"),
  numVar = "wage",
  weightVar = "wgt",
  hhId = "idh")
bgd05sdc


   ## The input dataset consists of 5000 rows and 41 variables.
   ## --> Categorical key variables: gender, age, marital, empstat, reg01
   ## --> Numerical key variables: wage
   ## --> Weight variable: wgt
   ## --> Cluster/Household-Id variable: idh
   ## ----------------------------------------------------------------------
   ##
   ## Information on categorical key variables:
   ##
   ## Reported is the number, mean size and size of the smallest category for
   recoded variables.
   ## In parenthesis, the same statistics are shown for the unmodified data.
   ## Note: NA (missings) are counted as seperate categories!
   ##
   ## Key Variable Number of categories Mean size
   ## gender 2 (2) 2500.000 (2500.000)
   ## age 93 (93) 53.763 (53.763)
   ## marital 5 (5) 1244.250 (1244.250)
   ## empstat 5 (5) 337.250 (337.250)
   ## reg01 7 (7) 714.143 (714.143)
   ## Size of smallest
   ## 2492 (2492)
   ## 1 (1)
   ## 39 (39)
   ## 82 (82)
   ## 0 (0)
   ## ----------------------------------------------------------------------
   ##
   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ## - 2-anonymity: 477 (9.540%)
   ## - 3-anonymity: 967 (19.340%)
   ## - 5-anonymity: 1801 (36.020%)
   ##
   ## ----------------------------------------------------------------------
   ##
   ## Numerical key variables: wage
   ##
   ## Disclosure risk is currently between [0.00%; 100.00%]
   ##
   ## Current Information Loss:
   ## - IL1: 0.00
   ## - Difference of Eigenvalues: 0.000%
   ## ----------------------------------------------------------------------
```

From the previous summary of the *sdcMicroObj* object, it can be seen that about 2% of the observations is violating 3-anonymity. Additionally, *age* is given in years and the smallest group consists of only one person. Of course, the risk of the continuous variables is high and its information loss is zero since no anonymization have been applied yet. We can also report the disclosure risk, e.g. based on the individual risk approach (Franconi and Polettini 2004b).

```
print(bgd05sdc, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the data:
   0
   ## Expected number of re-identifications: 2.17 (0.04 %)
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications: 3.16 (0.06 %)
   ## --------------------------------------------------------------------
```

One of the most important variables for the World bank is age. Thus it should only be touched with care. On the other hand, age has far the most categories. Let us have a closer look on age in Fig. 8.1. This figure can be produced with

```
## Code for Fig. 8.1
ggplot(bgd05, aes(x = age)) + geom_histogram(binwidth=1)
```

A histogram with bin-width, or in other words a bar-plot of age, is drawn. An age heaping effect can be observed and few values with age over 75. Note that R package simPop includes methods to lower the effect of age heaping. For the analysis of this data set it is important that the data utility of the working class is as high as possible. Top-coding is thus a very good candidate to aggregate the older population to one category. In the following this is applied for variable *age*.

To ensure 3-anonymity, local suppression is applied for the selected categorical key variables and microaggregation is applied for the continuous key variables. The necessary code is shown in the following.

```
## top-coding of age:
bgd05sdc <- topBotCoding(bgd05sdc,
                         value = 66,
                         replacement = 75,
                         kind = "top",
                         column = "age")
## ensuring 3-anonymity:
bgd05sdc <- localSuppression(bgd05sdc,
                         k=3,
                         importance = c(2,1,4,3,5))
```

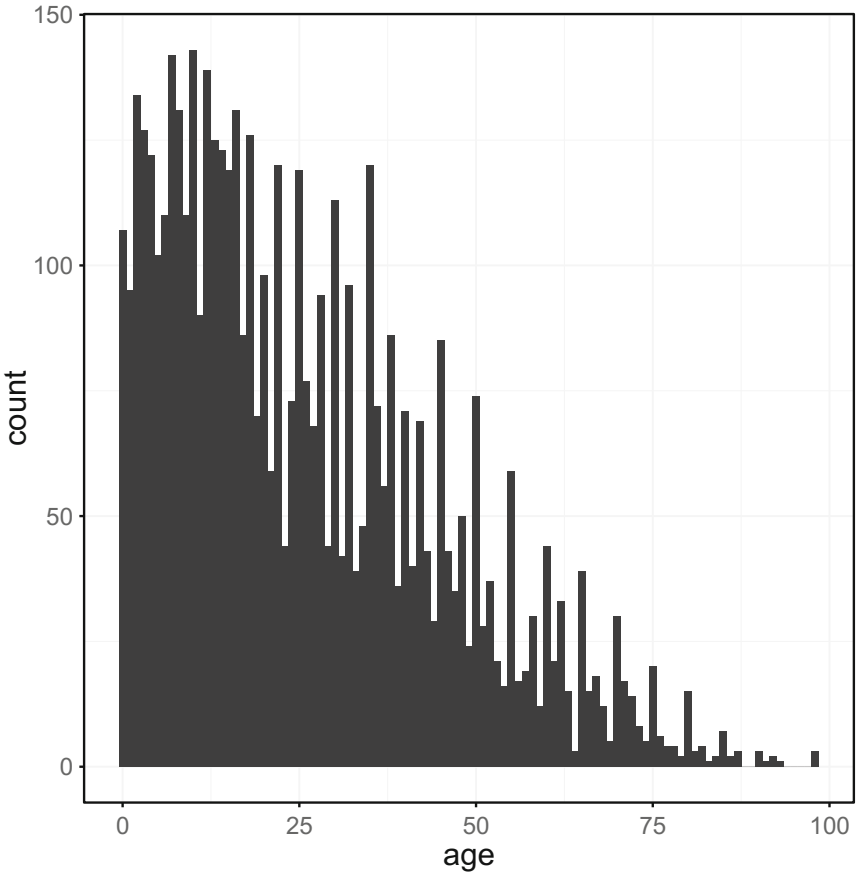


Fig. 8.1 Distribution of age in the Bangladesh 2005 I2D2 data set

In the next code block, the number of suppressions is shown.

```
print(bgd05sdc, "ls")

   ## Local suppression:
   ##   KeyVar | Suppressions (#) | Suppressions (%)
   ##   gender |                1 |            0.020
   ##      age |                2 |            0.040
   ##  marital |               59 |            1.180
   ##  empstat |                3 |            0.060
   ##    reg01 |              376 |            7.520
   ## ----------------------------------------------------------------------
```

We see that variable age is almost not effected from suppressions, which was our goal. The most suppressions are done in region (*reg01*). Region was chosen with the highest number of importance, since it is a variable that is anyhow often of not good quality in the I2D2 data. It follows that most suppressions are done in region.

Local suppressions ensures k-anonymity. We can check this in the following.

```
print(bgd05sdc, "kAnon")

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 0 (0.000%) | in original data: 477 (9.540%)
   ##   - 3-anonymity: 0 (0.000%) | in original data: 967 (19.340%)
   ##   - 5-anonymity: 747 (14.940%) | in original data: 1801 (36.020%)
   ##
   ## ----------------------------------------------------------------------
```

The risk can be shown from the anonymized data and compared with the risk from the original data (note that 3-anonymity is already reached after applying `localSuppression()`):

```
print(bgd05sdc, "risk")


   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the data:
   ##   in modified data: 0
   ##   in original data: 0
   ## Expected number of re-identifications:
   ##   in modified data: 0.31 (0.01 %)
   ##   in original data: 2.17 (0.04 %)
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications:
   ##   in modified data: 0.46 (0.01 %)
   ##   in original data: 3.16 (0.06 %)
   ## ----------------------------------------------------------------------
```

The risk reduced significantly. For the categorical variables it can be concluded that the risk reduced a lot, and the data set will have very low risk and high data utility.

In the next step we want to apply microaggregation to the continuous key-variable 'wage' that was chosen when specifying the disclosure risk scenario. To do so, we can apply function `microaggregation` as shown in the next code block, with aggregation level equals 4. This should provide enough anonymization since a data intruder cannot decide which observation is the correct link whenever a link can be made. If this is not accepted, then further perturbation can be made by adding noise or shuffling.

```
bgd05sdc <- microaggregation(bgd05sdc, aggr=4)
```

Note that exactly the same results can easily be produced using the **sdcMicroGUI** package. To import a data set one has to select `Data` → `Import` → `Import STAT` and choose the desired data set from your hard disk. After the data set has been processed, a window will automatically pop-up that allows to specify the key variables.

Table 8.3 Number of observations and percentage of observations that violating k-anonymity for data sets from Bangladesh, India, Senegal, Thailand, Vietnam and South Africa

	BGD05	IND99	SEN01	THA09	VNM06	ZAF07
viol2anon	539.00	2547.00	0.00	13678.00	879.00	377.00
viol3anon	1031.00	5484.00	0.00	26446.00	1691.00	1065.00
viol4anon	1532.00	8240.00	0.00	38200.00	2401.00	1924.00
viol2anon_perc	1.10	0.43	0.00	9.80	2.25	0.36
viol3anon_perc	2.11	0.93	0.00	18.95	4.33	1.01
viol4anon_perc	3.13	1.39	0.00	27.37	6.15	1.82

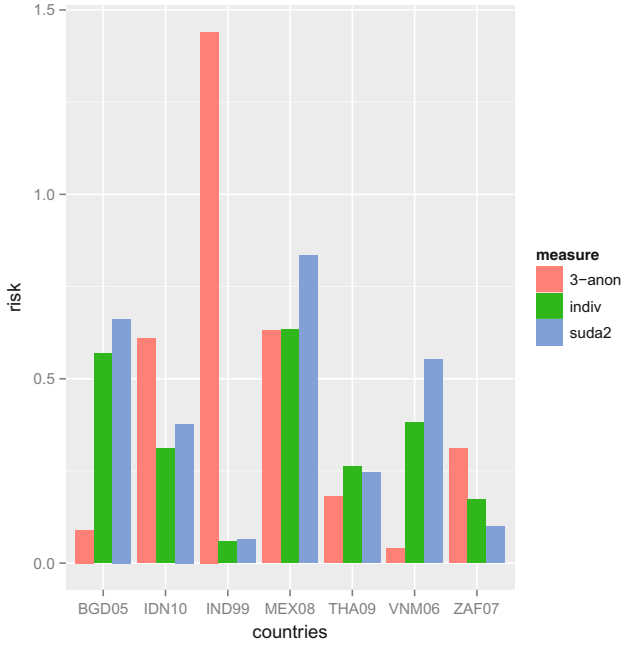


Fig. 8.2 Global disclosure risk for each country: percentages of observations violating 3-anonymity, percentage of observations with risk >0.01, percentage of observations with a suda score >0.01

8.4.4 Results for All Other Countries

Subsequently, the code shown in the previous chapter is applied to other country data sets.

Table 8.3 reports for each country the numbers and percentages of observations violating k-anonymity. In the table k runs from two to four.

Table 8.4 Number of strata and average size of strata

	BGD05	IDN10	IND99	MEX08	THA09	VNM06	ZAF07
strata	8280.00	48300.00	10350.00	27600.00	13800.00	4140.00	8280.00
cn	5.91	12.24	5.67	5.06	2.83	25.58	5.91

Table 8.5 Percentage of local suppressions

	Gender	Age	Marital	Empstat	Reg
BGD05	0.000	0.086	0.000	0.000	0.000
IDN10	0.000	0.175	0.001	0.000	0.007
IND99	0.000	0.017	0.003	0.000	0.000
MEX08	0.000	0.232	0.000	0.000	0.000
THA09	0.000	0.112	0.000	0.000	0.000
VNM06	0.000	0.141	0.000	0.000	0.000
ZAF07	0.000	0.024	0.000	0.000	0.000

Figure 8.2 displays the results of three different disclosure risk measures. The disclosure measures are:

- the percentage of observations violating 3-anonymity
- the percentage of observations having individual risk larger than 0.01
- the percentage of observations with suda score larger than 0.01

We can observe from this Figure that the risk-measures perform quite differently across countries.

In the following, the results after anonymization of all six available country data sets are shown.

The anonymization methods mentioned in Sect. 8.6.2 are applied to each country data set.

The top-coding of age results in larger sizes of this strata since *extreme* categories of the variable are combined. The amount of strata is reduced as it can be observed from Table 8.8. Another idea would be to categorize the variable *age* into age classes (which was not done in this case) using the function `globalRecode()` or the global recode facilities of the package **sdcMicroGUI**. This would result in much larger sizes of the strata (Table 8.4).

Only few values had to be suppressed by the suppression functions that were used according to Table 8.5. Naturally, the percentage of suppressions is slightly higher for countries with small strata—an example would be Mexico in 2008 (Fig. 8.3).

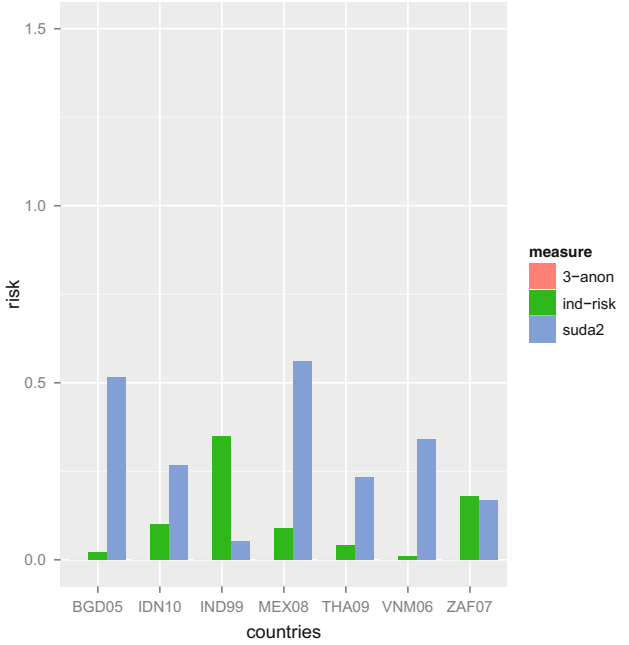


Fig. 8.3 Global disclosure risk for each country: percentages of observations violating 3-anonymity, percentage of observations with risk >0.01, percentage of observations with a suda score >0.01

According to Fig. 8.8, also the risks have been reduced greatly. The percentages of observations with individual risk >1% is almost zero for Bangladesh and Vietnam and only for India 1999, the risk might be reduced further by using additional anonymization methods such as recoding or swapping.

8.4.5 Data Utility

The data utility is not evaluated here, since it was not requested by World bank or the countries. Their aim was to provide anonymized data sets with minimal modifications. There was also no study, which indicators are most important as with the EU-SILC and SES data.

Table 8.6 Number of observations (obs) and number of variables (vars) for the investigated P4 data sets—household (HH) and individual (IND) file according to data sets from Mexico, Thailand and Tajikistan

	MEX10	THA09	TJK07
obs HH	27654	43844	4860
vars HH	60	60	61
obs IND	107779	139590	30323
vars IND	9	9	9

8.5 Anonymization of P4 Data

The P4 data is important to estimate purchase power parities (PPP). The P4 data consists of three files, one data set containing the information on households, another data set containing the information on individual level, and the PPP data set. Thus the anonymization of the P4 data is different to previous ones. Here the data are anonymized at household level as well as on individual level (Table 8.6).

For this case study we investigate data sets from three countries. The number of observations and the amount of variables for the household and individual data set are listed in Table 8.10.

Due to World Bank's contract and bilateral agreements, national statistical offices have authorized the use of these data sets for this case study.

8.5.1 Key Variables

It is necessary to be very careful while determining a disclosure scenario. In this case study three categorical variables on household level and four variables on individual level are selected.

8.5.1.1 Key Variables on Household Level

Variables determined as key variables for statistical disclosure control on household level are visible in Table 8.7. Additionally, the variable holding the sampling weights is printed. Moreover, the number of strata given by cross tabulation of the chosen three key variables is given.

Not many strata exist in Tajikistan (TJK) because all observations belong to the same defined region.

Table 8.7 Information on categorical key variables, sampling weights and information on strata sizes for the household information on the investigated three countries. The number in brackets indicates the number of categories

	MEX10	THA09	TJK07
key1	geo_2(600)	geo_2(76)	geo_2(1)
key2	rururb(2)	rururb(2)	rururb(2)
key3	hhsize(18)	hhsize(17)	hhsize(21)
weight	wta_hh	wta_hh	wta_hh
strata	21600	2584	42

Table 8.8 Information on categorical key variables, sampling weights and information on strata sizes for the household information on the investigated three countries. The number in brackets indicates the number of categories

	MEX10	THA09	TJK07
key1	sex(2)	sex(2)	sex(2)
key2	age(98)	age(100)	age(99)
key3	marital(7)	marital(7)	marital(9)
key4	geo_1(32)	geo_1(5)	geo_1(5)
weight	wta_hh	wta_hh	wta_hh
strata	43904	7000	8910

8.5.2 Key Variables on Individual Level

Variables determined as key variables for statistical disclosure control on individual level are visible in Table 8.8. Additionally, the variable holding the sampling weights is printed. Moreover, the number of strata given by cross tabulation of the chosen three key variables is given.

Attention: it is easy to see that the geo-coding for Tajikistan (TJK) is wrong. The finer geographical code, `geo_2`, is smaller as `geo_1`.

8.5.3 sdcMicro Code for One Example Country

In the following we exemplify show the required code to anonymize the P4 data from Mexico in 2010, see the code block below.

The aim is to ensure confidentiality on household and individual level. Therefore, the household data file gets anonymized first. Secondly the household information is merged to the individual file before the merged data set gets a final anonymization.

To start with, STATA household and individual data sets are imported in R. (see the first lines of the code block below. An *sdcMicroObj* is then created for the household

data by specifying important variables for statistical disclosure control. 3-anonymity on household level is guaranteed by local suppression. Having some basic anonymizations done on household level, the household and the individual data are merged together. On this file with individual and household information, *age* below 45 is categorized into five groups. To ensure 3-anonymity on individual level, local suppression is applied for the selected categorical key variables. The anonymization of the remaining high-risk observations can be achieved by using function `localSupp` and specifying a suitable threshold. Then the data are splitted again into a household and an individual file. Finally, the protected data are written on hard disk as STAT files, see for details, the necessary code in the code listed below.

```
### --- begin anonymization
## import STATA data sets into R:
hh <- read.dta("mex2010_hld.dta")
ind <- read.dta("mex2010_ind.dta")
## create sdcMicro object
mexhh <- createSdcObj(hh,
keyVars=c('geo_2','rururb','hhsize'),
weightVar="wta_hh")
## Step 1) 3-Anon for Households
mexhh <- localSuppression(mexhh, k=3)
hhout1 <- extractManipData(mexhh)
## Step 2) Merging the household information to the individuals
indhh <- merge(ind, hhout1, all.x=TRUE)
## Step 3) Doing some SDC on individuals
indhh$age[indhh$age > 45 & indhh$age <= 50] <- 47.5
indhh$age[indhh$age > 50 & indhh$age <= 55] <- 52.5
indhh$age[indhh$age > 55 & indhh$age <= 60] <- 57.5
indhh$age[indhh$age > 60 & indhh$age <= 65] <- 62.5
indhh$age[indhh$age > 65 ] <- 75
mexind <- createSdcObj(indhh,
keyVars=c('sex','age','marital','geo_1'),
hhId="hid", weightVar="wta_hh")
# 3-Anon for Households
mexind <- localSuppression(mexind, k=3)
# lower individual risk
mexind <- localSupp(mexind, keyVar="marital", threshold=0.9)
# extract anonymized data
dataM <- extractManipData(mexind)
## Step 4) Separating the household and the individuals part
hhout2 <- dataM[!duplicated(dataM$hid),colnames(hhout1)]
indout <- dataM[,colnames(ind)]
# write anonymized data as STATA file on hard disk
write.dta(hhout2,file="mex2010hh_anon.dta")
write.dta(indout,file="mex2010ind_anon.dta")
```

The anonymization of the remaining high-risk observations can be achieved by using function `localSupp` and specifying a suitable threshold.

After performing these steps it is easy to see that 3-anonymity has been achieved as shown below.

```
print(mexind)
```

```
Number of observations violating

 -  2-anonymity:  0 (orig:  77 )
 -  3-anonymity:  0 (orig:  156 )
--------------------------

Percentage of observations violating
 -  2-anonymity:  0 %  (orig:  0.07 % )
 -  3-anonymity:  0 %  (orig:  0.14 % )
```

By supplying an additional parameter to the print function we can see that the risk in the data is considerable lower after performing the previous anonymization steps as shown below:

```
print(mexind, "risk")
```

```
--------------------------
0  (orig: 0 ) obs. with higher risk than the main part
Expected no. of re-identifications:
 283.74 [ 0.26 %]  (orig: 794.62 [ 0.74 %])
--------------------------
--------------------------
Hierarchical risk
--------------------------
Expected no. of re-identifications:
 1388.95 [ 1.29 %]  (orig: 3569.15 [ 3.31 %])
```

We note that it is also very interesting to see that under the given disclosure scenario and the performed operations on the data set, no additional suppressions (suppressions after `localSuppression`: 0, 38, 17077, 0) occurred when local suppression (`localSupp(bgd05sdc, keyVar = "marital", threshold = 0.01)`) was applied to observations with possibly high risk. This means that the risk in the data set was already low enough since no observations had had individual risks higher than the chosen threshold value. Note that variable `marital` has already 24921 missing values included in the original file.

```
print(mexind, "ls")
```

```
sex ..... 0 [ 0 %]

age ..... 38 [ 0.035 %]

marital .. 17077 [ 15.844 %]

geo_1 .... 0 [ 0 %]
```

8.5.4 Results for All Other Countries

Subsequently, the code shown in the previous chapter is applied to other country data sets.

Table 8.13 reports for each country the numbers and percentages of observations violating k-anonymity. In the table k runs from two to four.

Figure 8.4 displays the results of three different disclosure risk measures. The disclosure measures are:

- the percentage of observations violating 3-anonymity
- the percentage of observations having individual risk larger than 0.01
- the percentage of observations with sudo score larger than 0.01

We can observe from this Figure that the different risk-measures perform quite different across countries (Table 8.9).

In the following, the results after anonymization of all available country data sets are shown.

Table 8.9 Observations violating k-anonymity

	MEX10	THA09	TJK07
viol2anon	77.00	186.00	295.00
viol3anon	156.00	400.00	545.00
viol4anon	255.00	688.00	698.00
viol2anon_perc	0.07	0.13	0.97
viol3anon_perc	0.14	0.29	1.80
viol4anon_perc	0.24	0.49	2.30

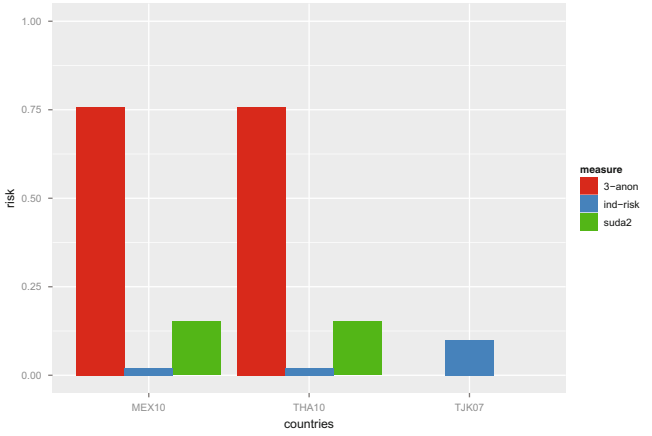


Fig. 8.4 Global disclosure risk for each country: percentages of observations violating 3-anonymity, percentage of observations with risk >0.01, percentage of observations with a suda score >0.01

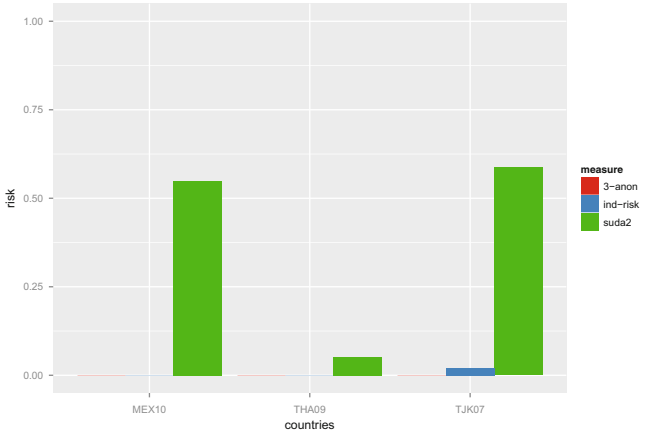


Fig. 8.5 Global disclosure risk for each country after local suppression

The anonymization methods mentioned in Sect. 8.6.2 are applied to each country data set.

According to Fig. 8.8, the risks have been reduced a lot (compare with Fig. 8.4 which shows the risk of the orginal data). The percentages of observations with individual risk $>1\%$ is almost zero for Bangladesh and Vietnam and only for India 1999, the risk might be reduced further by using additional anonymization methods such as recoding or swapping (Fig. 8.5).

8.6 Anonymization of the SHIP Data

The coding for this section was mainly done by Bernhard Meindl.

The SHIP data consist of three files. Most important, one data set containing the information on households, another data set containing the information on individual level.

For the case study we investigate a total of 3 data sets from 2 countries. The number of observations and the amount of variables for the household and individual data set are listed in Table 8.10.

Due to World Bank's contract and bilateral agreements, national statistical offices have authorized the use of these data sets for this case study.

Table 8.10 Number of observations (obs) and number of variables (vars) for the investigated SHIP data sets—household (hh) and individual (ind) file

	ETH04	GHA98	GHA05
obs HH	21595	5998	8687
vars HH	119	122	122
obs IND	99229	26411	37128
vars IND	43	92	92

Table 8.11 Information on categorical key variables, sampling weights and information on strata sizes for the household information on the investigated three data sets. The number in brackets indicates the number of categories

	ETH04	GHA98	GHA05
key1	district (31)	district (18)	district (18)
key2	rururb (2)	rururb (2)	rururb (2)
key3	hhsize (18)	hhsize (18)	hhsize (24)
weight	wta_hh	wta_hh	wta_hh
strata	1116	648	864

8.6.1 Key Variables

It is necessary to be very careful while determining a disclosure scenario. In this case study three categorical variables on household level and four variables on individual level are selected.

8.6.1.1 Key Variables on Household Level

Variables determined as key variables for statistical disclosure control on household level are visible in Table 8.11. Additionally, the variable holding the sampling weights is printed. Moreover, the number of strata given by cross tabulation of the chosen three key variables is given.

We can observe that the number of strata is quite different in the three household data sets.

8.6.1.2 Key Variables on Individual Level

Variables determined as key variables for statistical disclosure control on individual level are visible in Table 8.12. Additionally, the variable holding the sampling weights is printed. Moreover, the number of strata given by cross tabulation of the chosen three key variables is given.

Table 8.12 Information on categorical key variables, sampling weights and information on strata sizes for the household information on the investigated three data sets. The number in brackets indicates the number of categories

	ETH04	GHA98	GHA05
key1	sex (2)	sex (2)	sex (2)
key2	agey (98)	agey (99)	agey (99)
key3	marstat (5)	marstat (6)	marstat (6)
key4	region (10)	region (10)	region (10)
key5	ethnicity (0)	ethnicity (20)	ethnicity (51)
weight	wta_hh	wta_hh	wta_hh
strata	9800	237600	605880

From Table 8.12 it is easy to see that there are some problems in the data concerning the variable `'ETHNICITY'`. This variable is completely missing in the GHA98 data set. For this reason it is not sensible to use this variable as a key-variable. Therefore, `'ETHNICITY'` will be omitted in this data set only.

8.6.2 sdcMicro Code for One Example Country

We continue to show the necessary `R`-code to anonymize the SHIP data from Ghana in 2005 using the `R` package `sdcMicro`.

The aim is to ensure confidentiality on household and individual level. Therefore, the household data file gets anonymised first. In a second step the household information is merged to the individual file before the complete, merged data set gets a final anonymization.

8.6.2.1 Preparing the Data Sets

The first steps consists of reading STATA-input files into `R`. Using package `foreign` it is easy to do so using function `read.dta`. The necessary steps are:

(1) read both files containing household information and merge them together to obtain one single household-file (`hh`).
(2) read both files containing information on individuals and merge them together to obtain one single individual-file (`ind`).
(3) create an object of class `sdcMicroObj` given our choice of key-variables and weight-variable as discussed above.

These steps are listed in the following.

```
## step 1: read household information into R
# household and individual information are available in two files
# we need to merge them on common variables
hh1 <- read.dta("GHA_2005_E.dta")
hh2 <- read.dta("GHA_2005_H.dta")
hh <- merge(hh1, hh2, sort=F)

## step 2: read individual information into R
ind1 <- read.dta("GHA_2005_I.dta")
ind2 <- read.dta("GHA_2005_L.dta")
ind <- merge(ind1, ind2, sort=F)

## step 3: create sdcMicro object
obj.hh <- createSdcObj(dat=hh,
  keyVars=c("DISTRICT", "RURURB", "HHSIZE"),
  pramVars=c("ROOF","WALLS","FLOOR"),
  numVars=c("HSUTILITY", "GAS", "ELEC"),
  weightVar="WTA_HH")
```

8.6.2.2 Dealing with Variables on Household Level

Since a set of categorical key-variables has already been defined and we have already produced an sdcMicro object to work with (`obh.hh`), we continue and perform the following steps to anonymize these variables.

(1) guarantee 3-anonymity on household level by applying local suppression to the set of key-variables.
(2) apply post-randomization to another set of categorical variables. The variables `'ROOF'`, `'WALLS'` and `'FLOOR'` need to be protected but it is not necessary to define them as key-variables and also provide 3-anonymity for them.
(3) perform microaggregation on a set of numeric key variables.

These steps are shown in detail below.

```
## step 1: provide 3-anon for households
obj.hh <- localSuppression(obj.hh, k=3)

## step 2: apply pram to other categorical variables
obj.hh <- pram_strata(obj.hh)

## step 3) microaggregation and proportional adjusting
obj.hh <- microaggregation(obj.hh,
  variables="HSUTILITY", method="individual")
obj.hh <- prop.split(obj.hh, numVars=numVarsHH)
```

'obj.hh was created within the previous code and is an object of class 'sdcMicroObj. Thus we can apply implemented methods such as localSuppression or pram_strata to it. To use the default options it is not necessary to provide any input other than an object of class *sdcMicroObj* as it can be seen how the function pram_strata was called.

The application of microaggregation should be explained in more detail. In this example we used three numeric key-variables—'HSUTILITY', 'WALLS' and 'FLOOR'—as it was shown in the code blocks. However, these variables contain a certain structure. To be specific, 'HSUTILITY' can be calculated as the sum of variables 'WALLS' and 'FLOOR'. If we want to keep this logical connex it is not suggested to apply microaggregation to all these variables simultaneously. The approach we have taken in this example was to microaggregate only the total by setting the parameter 'variable' to 'HSUTILITY' when applying microaggregation. To keep the structure a custom R-function (prop.split()) which is not included in sdcMicro) was applied to the modified variable that calculated new values for variables 'WALLS' and 'FLOOR' based on the ratio in the original data. We note that the sdcMicroObj 'obj.hh' contains all the necessary data for this calculation.

8.6.2.3 Dealing with Variables on Individual Level

Once we have conducted the anonymization methods on household-level, we continue to work on individual data. To do so, we need to perform the following steps:

(1) combine the modified household file and the individual data files.
(2) create an object of class 'sdcMicroObj using function createSdcObj and the merged data set from the previous step.
(3) apply global recoding to variable 'AGEY' on the resulting, combined data set. In this case we will create five age-groups for individuals older than 45 years.
(4) guarantee 3-anonymity on individual level by applying local suppression for the selected categorical key variables on individual level.
(5) protect remaining high-risk observations by suppressing additional values in variable 'MARSTAT' using function localSupp and a suitable threshold.

```
## step 1: merge household information to individual-file
hh.out <- extractManipData(obj.hh)
ind <- merge(ind, hh.out, all.x=TRUE)

## step 2: recoding age
ind$AGEY[ind$AGEY > 45 & ind$AGEY <= 50] <- 47.5
ind$AGEY[ind$AGEY > 50 & ind$AGEY <= 55] <- 52.5
ind$AGEY[ind$AGEY > 55 & ind$AGEY <= 60] <- 57.5
ind$AGEY[ind$AGEY > 60 & ind$AGEY <= 65] <- 62.5
ind$AGEY[ind$AGEY > 65 ] <- 75

## step 3: create an object of class 'sdcMicroObj'
obj.ind <- createSdcObj(dat=ind,
  keyVars=c("SEX", "AGEY", "MARSTAT", "REGION"),
  hhId="HID", weightVar="WTA_HH")

## step 4: provide 3-anon on individual-level
obj.ind <- localSuppression(obj.ind, k=3)

## step 5: deal with remaining high-risk observations
obj.ind <- localSupp(obj.ind, keyVar="MARSTAT", threshold=0.8)
```

8.6.2.4 Finalizing the Results

After performing all the steps described in the previous code listings, we can finalize the anonymization procedure and save the output files.

(1) splitting the complete file into household and individual file
(2) write anonymized files to hard disk

This is shown in the code below.

```
## step 1: split files
dataM <- extractManipData(obj.ind)
hh.anon <- dataM[!duplicated(dataM$HID), colnames(hh)]
ind.anon <- dataM[, colnames(ind)]

## step 2: write anonymised data as STATA file to hard disk
write.dta(hh.anon, file="GHA_2005_HH_anon.dta")
write.dta(ind.anon, file="GHA_2005_IND_anon.dta")
```

8.6.2.5 Results of the Anonymization Strategy

After performing these steps we now have a look at the results of the anonymisation procedure. It is easy to see that 3-anonymity has been achieved as it is shown from the following output.

```
print(ghaind, "kAnon")

   ## Infos on 2/3-Anonymity:
   ##
   ## Number of observations violating
   ##   - 2-anonymity: 0 (0.000%) | in original data: 686 (1.848%)
   ##   - 3-anonymity: 0 (0.000%) | in original data: 1455 (3.919%)
   ##   - 5-anonymity: 1233 (3.321%) | in original data: 2817 (7.587%)
   ##
   ## ------------------------------------------------------------------------
```

By supplying an additional parameter to the `print` function, we observe that due to the anonymization methods applied the remaining risk in the data is considerable lower than as it was in the original, unmodified data. We see that the number of expected reidentifications has decreased significantly.

```
   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main part of the data:
   ##   in modified data: 0
   ##   in original data: 0
   ## Expected number of re-identifications:
   ##   in modified data: 4.23 (0.01 %)
   ##   in original data: 14.26 (0.04 %)
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications:
   ##   in modified data: 26.93 (0.07 %)
   ##   in original data: 83.58 (0.23 %)
   ## ------------------------------------------------------------------------
```

Also from Fig. 8.6, which shows the hierarchical risks for all individuals, it can be concluded that the risk is very small for any observation.

8.6.3 Results for All Other Countries

Subsequently, the code shown in the previous chapter is applied to other country data sets.

Table 8.13 reports for each country the numbers and percentages of observations violating k-anonymity. In the table k runs from two to four.

Figure 8.4 displays the results of three different disclosure risk measures. The disclosure measures are:

- the percentage of observations violating 3-anonymity (`'3-anon'`)
- the percentage of observations having individual risk larger than 0.01 (`'ind-risk'`)
- the percentage of observations with suda score larger than 0.01 (`'suda2'`)

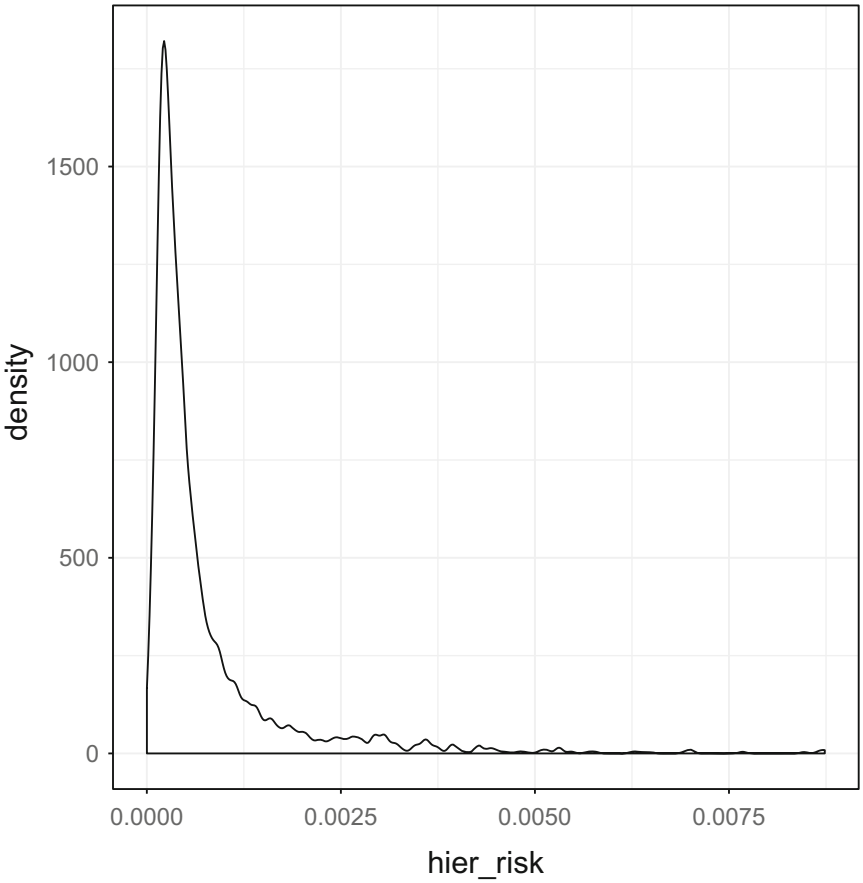


Fig. 8.6 Hierarchical risk for the anonymized SHIP data from Ghana after anonymisation

Table 8.13 Observations violating *k*-anonymity

	ETH04	GHA98	GHA05
viol2anon	65.00	69.00	74.00
viol3anon	171.00	129.00	146.00
viol4anon	294.00	216.00	224.00
viol2anon_perc	0.30	1.15	0.85
viol3anon_perc	0.79	2.15	1.68
viol4anon_perc	1.36	3.60	2.58

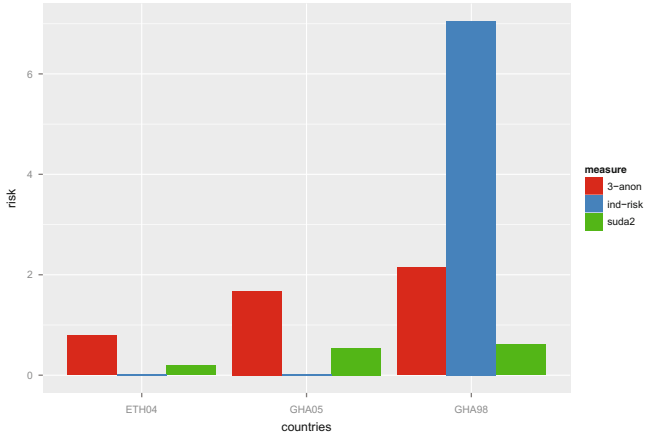


Fig. 8.7 Global disclosure risk measures based on original (unmodified) SHIP data sets from different countries

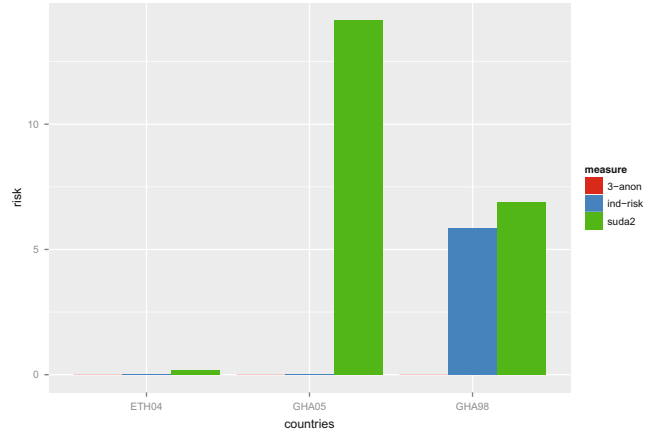


Fig. 8.8 Global disclosure risk measures based on protected (modified) SHIP data from different countries

We can observe from this Figure that the risk-measures perform quite differently across the data sets that have been considered (Fig. 8.7).

In the following, the results after anonymisation of all available 3 data sets are shown. The anonymisation methods discussed in Sect. 8.6.2 were applied to each data set.

According to Fig. 8.8, the risks have been reduced a lot. The percentages of observations with individual risk $>1\%$ is almost zero for `'ETH04'` and `'GHA05'`

and only for `'GHA98'`, the risk could be reduced even further by using additional anonymisation methods such as recoding or swapping.

8.6.3.1 Validity of the Anonymised Data Sets

The data utility is not evaluated here, since first it was not requested by the World bank or the countries and second we already know how to evaluate the data utility since it was already shown for the other data sets in this chapter. The aim of the World bank was to provide anonymized data sets with minimal modifications. There was also no study, which indicators are most important as with the EU-SILC and SES data.

However, of course hundreds of different results could be compared and analyzed to see if the anonymised data sets still have high utility. Instead of comparing dozens of tables and means on domains, it is also possible to analyse the impact of anonymization procedures by statistical modelling. In this case we would fit the same regression model containing modified variables (for example using one of the numeric key variables as the dependent variable) to both the original and the anonymized data set. Then we can quantify the impact of the anonymization procedure by comparing different results of the models.

8.7 A Synthetic Socio-economic Population and Sample

The simulation of a countries' population characteristics including information on demographic and economic variables is reported. The population is produced on basis of the raw EU-SILC data using synthetic data simulation methods from Chap. 6. The corresponding code is embedded in this chapter. Detailed instructions are given how to carry out each of the four involved data generation steps. In addition, the data utility is discussed. This section summarizes the supplementary material of Templ et al. (2017) and the work done in a special framework project (*SGA on PUF*) funded by Eurostat (project coordination, CBS, Peter-Paul de Wolf) and partners from INSEE (Maxime Bergeat) and DeStatis (Lydia Spies).

For the purpose of simulating synthetic data we use the R package **simPop** (Templ et al. 2017) to simulate fully synthetic population data. A lot of major variables of SILC data set consist of continuous variables, mainly information about individual and household income. With synthetic data it is possible to keep the household structure in the public use files as well as typical applied traditional methods like global recoding and local suppression are not necessary to apply and thus we gain data with full details on all simulated variables.

As described in Chap. 6, the data simulation framework consists of four main steps:

1. setup of the household structure (see also Sect. 6.2.1);

2. Simulation of categorical variables (see also Sect. 6.2.1);
3. Simulation of (semi-)continuous variables (see also Sect. 6.2.1);
4. Splitting (semi-)continuous variables into components (see also Sect. 6.2.1).

Before the simulation some data preprocessing is necessary.

The *European Union Statistics on Income and Living Conditions* (EU-SILC) is a panel survey conducted in European countries and serves as data basis for the estimation of social inclusion indicators (cf. Atkinson et al. 2002) in Europe. EU-SILC data are highly complex and contain detailed information on the income of the sampled individuals and households.

Table 8.14 lists and describes some of the EU-SILC variables used for generating the synthetic population. Some categories of economic status and citizenship, respectively, will be later on combined due to their low frequency of occurrence; the combined categories are marked with an asterisk (∗). The variables *hsize*, *age* and *netIncome* are not included in the original EU-SILC data set. *hsize* has been computed by counting the number of persons in each household, *age* is derived from the year of birth, and *netIncome* is the sum of the income components. A complete description of EU-SILC variables can be found in Eurostat (2004) or newer versions of this document.

8.7.1 Data Preprocessing

8.7.1.1 Import of Data and Selection of Variables

The data provided by the SILC group at Statistics Austria contains the merged Austrian SILC information in SPSS. To read the data it may convenient to install package **haven** first.

For data import and bringing them to the right structure we use functions `loadSILC`, `mergeSILC` and `chooseSILC`. We do not go into details here but we refer to the help pages of these functions that are available in package **simPop**. We can use the utility function `?loadSILC` (available in package **simPop**) to import your data, where `orig_file` defines the path to your data set that can be stored in different formats. Note that this code cannot be reproduced by all readers of the book, since this raw data set cannot be distributed because of confidential reasons. It is only possible for readers who working at a NSI with access to the raw data. Nevertheless, we list all the code and one can also learn by simply looking at the code listed in this section of the book.

Table 8.14 Some important variables (out of many hundreds of variables) of the Austrian EU-SILC data. Originally published in Templ et al. (2017). Published with kind permission of ©Matthias Templ, Bernhard Meindl, Alexander Kowarik and Oliver Dupriez 2017 under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium provided the original author(s) and source are credited

Variable	Name	Possible outcomes	
Region	*db040*	1	Burgenland
		2	Lower Austria
		3	Vienna
		4	Carinthia
		5	Styria
		6	Upper Austria
		7	Salzburg
		8	Tyrol
		9	Vorarlberg
Household size	*hsize*		Number of persons in household
Age	*age*		Age (for the previous year) in years
Gender	*rb090*	1	Male
		2	Female
Self-defined current economic status	*pl030*	1	Working full-time
		2	Working part-time
		3	Unemployed
		4	Pupil, student, further training, or unpaid work experience or in compulsory military or community service*
		5	In retirement or in early retirement or has given up business
		6	Permanently disabled or/and unfit to work or other inactive person*
		7	Fulfilling domestic tasks and care responsibilities
Citizenship	*pb220a*	1	Austria
		2	EU*
		3	Other*
Personal net income	*netIncome*		Sum of income components listed below
Employee cash or near cash income	*py010n*	0	No income
		>0	Income

(*continued*)

Table 8.14 (*continued*)

Variable	Name	Possible outcomes	
Cash benefits or losses from self-employment	*py050n*	<0	Losses
		0	No income
		>0	Benefits
Unemployment benefits	*py090n*	0	No income
		>0	Income
Old-age benefits	*py100n*	0	No income
		>0	Income
Survivor's benefits	*py110n*	0	No income
		>0	Income
Sickness benefits	*py120n*	0	No income
		>0	Income
Disability benefits	*py130n*	0	No income
		>0	Income
Education-related allowances	*py140n*	0	No income
		>0	Income

* combined categories

Note that in some countries the data set is stored in four different files separating individual and household data (dfile, hfile, pfile, rfile). In this case we first need to merge the four files to one single file.

```
  filed <- "zielvar_d_eurostat2013.sav"
  filer <- "zielvar_r_eurostat2013.sav"
  filep <- "zielvar_p_eurostat2013.sav"
  fileh <- "zielvar_h_eurostat2013.sav"
  suf4 <- loadSILC(filed = filed, filer = filer, filep = filep,
  fileh = fileh)
```

```
suf4 <- loadSILC(filed = filed,
                 filer = filer,
                 filep = filep,
                 fileh = fileh)
```

Next data are merged to one file.

```
suf <- mergeSILC(d = suf4[["d"]],
                 r = suf4[["r"]],
                 h = suf4[["h"]],
                 p = suf4[["p"]])
```

Check which variable of the SUF are not present in the original data. Note this happens if countries store their data not using standards for EU-SILC.

```
cn <- checkCol(x, suf)

   ##
   ##  these variables are in y but not in x:
   ##  rx030 pw005_f
```

In the following, some variables are needed which might not be present for all countries. For the Austrian data, e.g., we add

```
  ## rx020, px020 age at the end of the income reference period
  suf$rx020 <- getAge(suf$rb080, year)
  suf$px020 <- suf$rx020
  ## household identification number
}
```

We extract the most important variables and convert some variables to the appropriate format if necessary.

```
eusilc <- chooseSILCvars(x, country = "Austria")

   ## Note: number of categories in pe040 was too small ( count = 1 ).
   ##  Categories 0 and 1 in pe040 have been combined to category 0-1
```

8.7.1.2 Adding Some Useful Variables and Minor Modifications

Various useful helper functions are written and included in simPop package. Application of these utility functions can not be seen in the following, but is avaiable when typing `?getAge`.

Following modifications are done.

- In the EU-SILC data set, only the birth is given. We add the age variable.
- The gender is coded with 1's and 2's. It is changed to *female* and *male*.
- The household size is only indirectly included in original data set (through the household ID), but we calculate and add it in the original data set.

- We slightly modify `pl031` (economic status). Variable `pb220a` (citizenship) is very sparse. We make broader categories. Note that this is country-specific.
- Occupation (`pl051` variable) and NACE (`pl111` variable) are very detailed in original microdata. Given these important variables are then simulated, we need to make global recoding to make broader categories. How the recoding has to be done is country specific. Recoding of Occupation to 1-digit version.
- Recoding of NACE code—this may be country-specific.
- Further, we need to compute the summarized personal gross income as well as the summarized household incomes. We choose to use the gross income variables instead of the net income variables because the equivalized disposable income is computed from the gross income variables. Moreover the net income variables are not compulsory and therefore some countries do not transmit them.
- For latter simulation use, we need the categorized incomes as well because they are used in the simulation described in next section. The gross income categories are constructed internally with the function `getBreaks()`, where default breakpoints based on quantiles are computed. Quantiles are based on the original weighted sample. In this example, the argument `upper` is set to `Inf` to avoid problems with different maximum values in the three synthetic populations, and, internally, the argument `equidist` is set to `FALSE` such that non-equidistant probabilities as described in Alfons et al. (2011) are used for the calculation of the quantiles. The household income is separated between "positive income" called `hgrossIncome` and "negative income" paid by the household, e.g. taxes, called `hgrossminus`.

```
eusilc <- modifySILC(x = eusilc, country = "Austria")
breaks <- c(min(eusilc$age, na.rm = TRUE), seq(15, 65, 15),
            max(eusilc$age, na.rm=TRUE))
eusilc$ageCat <- cut(eusilc$age,
                     breaks=breaks, include.lowest=TRUE)
```

For simulation purposes we need to include an extra level for the missing values (`NA`) in the categorical variables we are going to simulate later on in Sect. 8.7.2.

```
vars <- c("pl031", "pb220a", "pb190", "pl051", "pe040", "pl111")
for(i in vars){
  eusilc[, i] <- factorNA(eusilc[, i], newval = "NAs")
}
```

The data set is now prepared to simulate synthetic data. Some of the most important variables can again be seen in the following.

```
str(eusilc)
  ## `data.frame': 13250 obs. of 45 variables:
  ## $ db030 : num 1 1 2 2 2 2 2 3 3 3 ...
  ## $ db040 : Factor w/ 9 levels "Burgenland","Lower Austria",..: 6 6 4
  4 4 4 4 3 3 3 ...
  ## $ rb030 : num 101 102 201 202 203 204 205 301 302 303 ...
  ## $ rb080 : num 1990 1933 1969 1974 1995 ...
  ## $ rb090 : Factor w/ 2 levels "male","female": 1 2 2 1 2 2 1 1 2 2
  ...
  ## $ pl031 : Factor w/ 12 levels "1","10","11",..: 1 9 9 7 7 12 12 1 1
  8 ...
  ## $ pb220a : Factor w/ 4 levels "AT","EU","NAs",..: 1 1 1 1 1 3 3 2 2
  2 ...
  ## $ py010g : num 13556 0 0 0 1473 ...
  ## $ py021g : num 0 0 0 0 0 NA NA 0 0 0 ...
  ## $ py050g : num 0 0 0 0 0 NA NA 0 0 0 ...
  ## $ py080g : num 0 0 0 0 0 NA NA 0 0 0 ...
  ## $ py090g : num 0 0 0 0 8380 ...
  ## $ py100g : num 0 5284 0 0 0 ...
  ## $ py110g : num 0 4265 0 0 0 ...
  ## $ py120g : num 0 0 0 0 40 NA NA 0 0 0 ...
  ## $ py130g : num 0 0 22384 0 0 ...
  ## $ py140g : num 0 0 0 0 0 NA NA 0 0 0 ...
  ## $ hy040g : num 0 0 0 0 0 0 0 0 0 0 ...
  ## $ hy050g : num 0 0 9248 9248 9248 ...
  ## $ hy060g : num 0 0 0 0 0 0 0 0 0 0 ...
  ## $ hy070g : num 0 0 632 632 632 632 632 0 0 0 ...
  ## $ hy080g : num 0 0 0 0 0 0 0 0 0 0 ...
  ## $ hy090g : num 117 117 120 120 120 ...
  ## $ hy100g : num 0 0 0 0 0 0 0 0 0 0 ...
  ## $ hy110g : num 0 0 4164 4164 4164 ...
  ## $ hy120g : num 0 0 0 0 0 0 0 0 0 0 ...
  ## $ hy130g : num 0 0 0 0 0 0 0 720 720 720 ...
  ## $ hy140g : num 2364 2364 333 333 333 ...
  ## $ db090 : num 379 379 1051 1051 1051 ...
  ## $ rb050 : num 379 379 1051 1051 1051 ...
  ## $ pb190 : Factor w/ 6 levels "1","2","3","4",..: 1 4 2 2 1 6 6 1 1 1
  ...
  ## $ pe040 : Factor w/ 7 levels "0-1","2","3",..: 3 2 2 3 2 7 7 5 3 3
  ...
  ## $ pl051 : Factor w/ 10 levels "0-1","2","3",..: 7 6 9 7 5 10 10 2 3
  10 ...
  ## $ pl111 : Factor w/ 14 levels "a","b-e","f",..: 3 10 10 10 10 10 10
  7 3 10 ...
  ## $ rb010 : Factor w/ 1 level "2013": 1 1 1 1 1 1 1 1 1 1 ...
  ## $ age : num 22 79 43 38 17 15 13 49 47 18 ...
  ## $ hsize : Factor w/ 8 levels "1","2","3","4",..: 2 2 5 5 5 5 5 3 3 3
  ...
  ## $ pgrossIncome : num 13556 9549 22384 0 9893 ...
  ## $ hgrossIncome : num 117 117 14164 14164 14164 ...
  ## $ hgrossminus : num 2364 2364 333 333 333 ...
  ## $ pgrossIncomeCat: Factor w/ 12 levels "0","(0,463]",..: 6 5 7 1 5 1
  1 11 9 3 ...
  ## $ hgrossIncomeCat: Factor w/ 12 levels "0","(0,9.31]",..: 5 5 10 10
  10 10 10 7 7 7 ...
  ## $ hgrossminusCat : Factor w/ 22 levels "[-1.87e+03,-1.57e+03)",..:
  15 15 13 13 13 13 13 21 21 21 ...
  ## $ country : Factor w/ 1 level "Austria": 1 1 1 1 1 1 1 1 1 1 ...
  ## $ ageCat : Factor w/ 5 levels "[-1,15]","(15,30]",..: 2 5 3 3 2 1 1
  4 4 2 ...
```

8.7.2 Simulation of the Population

First, it is important to check the platform since of parallel computing tasks. Windows cannot fork and it is slow for parallel computing with data set of moderate or large size. Thus if the platform is Microsoft Windows we use only one CPU, otherwise the number of CPU's are set to maximum of CPU's on your computer minus one.

```
## simPop is parallel. The maximum number of CPU's minus one is used for
computations
## On the machine used to generate this document, the number of parallel
processes used is 15
```

After the main simulation described in this section a sample is selected from the synthetic population and some other variables can then be simulated in the resulting sample with a simpler method. The method is based on simulation of a population. One of the reasons for that is because we use logistic multinomial regression models in the simulation process and it is not easy to take sampling weights into account when building regression models. We finally produce also a sample just by sampling from the population.

In case of large populations (like Germany or France), the division of the household weight (=`db090`) by a factor, e.g. 10, is needed if the memory size on the computer is too small. Using a factor of 10 means that a population of size $N/10$, with N the number of inhabitants in the corresponding country, is simulated. We use the factor `myfactor = 1`. A reduced size of the population basically does not corrupt the population, but the population is no longer of the real size of the country. Using a server with about 32 GB of memory, also full populations of, e.g. Germany, can be produced.

```
eusilc$db090 <- eusilc$db090 / myfactor
```

8.7.2.1 Setup of the Household Structure

We specify the input and provide information on household ID, household size, strata (optionally) and sampling weights before simulation of a synthetic population with some basic variables. This can be all defined in function `specifyInput`, which produces an object of a certain class for that several methods are defined.

```
inp <- specifyInput(data=eusilc,
                    hhid="db030",
                    hhsize="hsize",
                    strata="db040",
                    weight="db090")
inp

   ##
   ##  --------------
   ## survey sample of size 13250 x 46
   ##
   ##  Selected important variables:
   ##
   ##  household ID: db030
   ##  personal ID: pid
   ##  variable household size: hsize
   ##  sampling weight: db090
   ##  strata: db040
   ##  --------------
```

The first step of the analysis is to set up the basic household structure using the function `simStructure()`. The argument `additional` specifies the variables that define the household structure in addition to the household size. We have chosen to generate in this step the age and gender variables. The result is an object of class *simPop* for that we subsequentially can apply functions like `simCategorical`, `simContinous` and many more.

```
eusilcP <- simStructure(inp, method="direct",
                    basicHHvars=c("age", "rb090") )
eusilcP

   ##
   ## --------------
   ## synthetic population  of size
   ##  8368615 x 7
   ##
   ## build from a sample of size
   ## 13250 x 46
   ## --------------
   ##
   ## variables in the population:
   ## db030,hsize,db040,age,rb090,pid,weight
```

The methodology for simulating these variables is:

1. define in original sample data stratum defined by combination of household size variable and variable defined in `strata` argument—we use region;
2. create randomly a population that has the same distribution for the household size and region variables;

Table 8.15 Variables simulated when generating the household structure using `simStructure`

Variable	Name	Type
Region	`db040`	Categorical
Household size	`hsize`	Categorical
Age	`age`	Categorical
Gender	`rb090`	Categorical
Sampling weight	`db090`	Continuous

3. for each simulated household, select randomly in the original microdata one household of the same region and the same size;
4. all members of the simulated household get the same individual characteristics assigned (variables defined in `additional` argument and sampling weight) as like the the original one.

Sampling weights are taken into account to reflect the distribution of region, household size, age and gender variables in the synthetic population.

Variables generated so far for the synthetic population are listed in Table 8.15.

8.7.2.2 Simulation of Categorical Variables

Additional categorical variables can be simulated for a synthetic population using the function `simCategorical`. Method `additional` specifies the variables to be simulated in this step via the function argument `regModel` that is used to specify the predictors and/or the regression model. Possible values are `basic` (only the basic structural variables are used as a predictor), `available` (all available predictors are used) or an object of class `formula` (specifying a formula). We choose `"available"`.

```
eusilcP <- simCategorical(eusilcP,
                          additional = c("pl031","pb220a", "pb190",
                                         "pl051", "pe040", "pl111"),
                          regModel = rep("available", 6),
                          nr_cpus=cpus, MaxNWts=5000 )
## for latter use:
age <- as.numeric(as.character(pop(eusilcP)$age))
breaks <- c(min(age, na.rm = TRUE), seq(15, 65, 15), max(age, na.rm=TRUE))
ageCat <- cut(age, breaks=breaks, include.lowest=TRUE)
pop(eusilcP, var="ageCat") <- ageCat
```

Table 8.16 Categorical variables simulated with `simCategorical`

Variable	Name	Type
Economic status	`pl031`	Categorical
Citizenship	`pb220a`	Categorical
Marital status	`pb190`	Categorical
Education	`pe040`	Categorical
Occupation	`pl051`	Categorical
Nace	`pl111`	Categorical

In this step, six categorical variables listed in the `additional` argument are simulated. The methodology for the simulation is as follows:

1. use of multinomial logistic regression modelling in order to estimate the distribution of the variable to be simulated—note that also regression trees and random forests can be used and selected in function `simCategorical`;
2. the response variable is the variable to be simulated. Explanatory variables are the household size, simple variables generated with the household structure (here age and gender) and already generated categorical variables;
3. in out case the order of simulation is important. In this case, `pl031` is an explanatory variable for simulation of `pb220a`, `pl031` and `pb220a` are used for simulation of `pb190` variable and so on.
4. logistic regression models are fitted for each stratum independently—the strata is defined for the whole simulation process and specified already in `specifyInput()`;
5. after estimation of the conditional distribution of the response variable, the values of the simulated variable are filled by random draws from this conditional distribution.

Variables generated in the synthetic population via function `simCategorical` are listed in Table 8.16.

8.7.2.3 Simulation of Continuous Variables

The continuous part of EU-SILC consists of information about income. We simulate gross income (and afterwards gross income components). The first step involves to compute total personal and household income. For household income positive and negative components are separated.

We use the function `simContinuous()` to simulate total income variables according to this methodology:

- use of multinomial logistic regression modelling (see Chap. 6) to estimate the distribution of the variable to be simulated;
- the response variable is a discretized version of the continuous variable to be simulated—and the option `zeros = TRUE` is used in order to add a special category for zero values for semi-continuous variables;
- explanatory variables are variables generated with the household structure (here age and gender—`rb090`) and all previously variables defined in the `regModel` argument—here we use occupation and citizenship, and the household size is used as well when simulating household income;
- logistic regression models are fitted for each stratum independently. The stratum is defined for the whole simulation process via the function `specifyInput`: we use here the region (`db040`) variable;
- after estimation of the conditional distribution of the response variable, random draws from this conditional distribution are taken to simulate the discretized version of the continuous variable;
- random draws from a generalized Pareto distribution (for the highest income category) or from uniform distributions (for all other categories) are realized;
- the `byHousehold = TRUE` option is used to ensure that every member of a household has the same household income.

```
breaks <- getBreaks(eusilc$pgrossIncome, eusilc$rb050,
                    upper = Inf, equidist = FALSE, zeros = TRUE)
breakshh <- getBreaks(eusilc$hgrossIncome, eusilc$rb050,
                    upper = Inf, equidist = FALSE, zeros = TRUE)
breakshhm <- getBreaks(eusilc$hgrossminus, eusilc$rb050,
                    upper = Inf, equidist = FALSE, zeros = TRUE)
```

Table 8.17 Variables simulated with `simContinuous`

Variable	Name	Type
Total personal income	`pgrossIncome`	Semi-continuous
Household income (total of positive components)	`hgrossIncome`	Semi-continuous
Household income (total of negative components)	`hgrossminus`	Semi-continuous

```
eusilcP <- simContinuous(eusilcP, breaks = breaks,
                         additional = "pgrossIncome",
                         upper = 200000, equidist = FALSE,
                         zeros=TRUE, nr_cpus=cpus, MaxNWts=10000,
                         regModel = formula(~ age + rb090 +
                         pl031 + pb220a))

## truncate personal income
p <- pop(eusilcP)$pgrossIncome
m <- max(samp(eusilcP)$pgrossIncome)
p[p > m] <- m
pop(eusilcP, var="pgrossIncome") <- p

eusilcP <- simContinuous(eusilcP, breaks = breakshh,
                         additional = "hgrossIncome",
                         upper = 200000, equidist = FALSE, zeros=FALSE,
                         nr_cpus=cpus, MaxNWts=10000,
                         regModel = formula(~ hsize + age + rb090 +
                         pl031 + pb220a),
                         byHousehold = TRUE)
## truncate hh income
p <- pop(eusilcP)$hgrossIncome
m <- max(samp(eusilcP)$hgrossIncome)
p[p > m] <- m
pop(eusilcP, var="hgrossIncome") <- p

eusilcP <- simContinuous(eusilcP, breaks = breakshhm,
                         additional = "hgrossminus",
                         upper = 200000, equidist = FALSE, zeros=FALSE,
                         nr_cpus=cpus, MaxNWts=10000,
                         regModel = formula(~ hsize + age + rb090 +
                         pl031 + pb220a),
                         byHousehold = TRUE)
```

Variables simulated in this step are listed in Table 8.17.

8.7.2.4 Simulation of Income Components

Total income variables are then splitted into income components with the `simComponents` function. Simulation of income components is based on the following methodology:

- the estimation is made for each group defined with the variable (argument `conditional`). For personal income we form groups according to economic status (`pl031` variable). For household income the household size is used to form the groups. These variables are already simulated in previous steps for the synthetic population.
- for each record of the population inside one group, select randomly a record from the original sample that has the same characteristics for the `conditional` variable. Sampling weights are taken into account to select the donor record.
- for the simulated observation impute all income components proportions from the original observation;

- any income component that consist of only missing values are not to be simulated (this will lead to an error).

```
if(country == "France"){
  ## in France "py021g" is missing
  pcomponents <- c("py010g","py050g","py080g",
                   "py090g","py100g", "py110g","py120g",
                   "py130g","py140g")

} else {
  pcomponents <- c("py010g","py021g","py050g","py080g",
                     "py090g","py100g", "py110g","py120g",
                     "py130g","py140g")
}

eusilcP <- simComponents(eusilcP,
                           total = "pgrossIncome",
                           components = pcomponents,
                           conditional = c("pl031","ageCat"))
```

In order to get the same household income components for all household members we create household data sets. This is done by just keeping the first household member of every household in the sample and in the population data.

```
hcomponents <- c("hy040g", "hy050g", "hy060g", "hy070g",
                 "hy080g", "hy090g", "hy110g")
eusilcP <- simComponents(eusilcP,
                         total = "hgrossIncome", components = hcomponents,
                         conditional = c("hsize", "db040"))

hminuscomponents <- c("hy120g","hy130g","hy140g")
eusilcP <- simComponents(eusilcP,
                         total = "hgrossminus",
                         components = hminuscomponents,
                         conditional = c("hsize","db040"))
```

The synthetic population has the following variables included at this moment:

```
eusilcP

   ##
   ## --------------
   ## synthetic population of size
   ## 8368615 x 40
   ##
   ## build from a sample of size
   ## 13250 x 46
   ## --------------
   ##
   ## variables in the population:
   ##
   db030,hsize,db040,age,rb090,pid,weight,pl031,pb220a,pb190,pl051,pe040,
   pl111,ageCat,pgrossIncomeCat,pgrossIncome,hgrossIncomeCat,hgrossIncome,
   hgrossminusCat,hgrossminus,py010g,py021g,py050g,py080g,py090g,py100g,
   py110g,py120g,py130g,py140g,hy040g,hy050g,hy060g,hy070g,hy080g,hy090g,
   hy110g,hy120g,hy130g,hy140g
```

We may add additional useful variables and we may also do some preparative modifications, e.g. changing the data type of specific variables.

```
if (is.factor(suf$py010g)) {
    require("dplyr")
    fac <- c("py010g", "py021g", "py050g", "py080g", "py090g", "py100g",
        "py110g", "py120g", "py130g", "py140g", "hy080g", "rb050",
        "pb040")
    suf <- suf %>% mutate_each_(funs(as.character), fac) %>%
    mutate_each_(funs(as.numeric), fac)
}

suf$pgrossIncome <- rowSums(suf[, c("py010g", "py021g", "py050g",
    "py080g", "py090g", "py100g", "py110g", "py120g", "py130g", "py140g")],
    na.rm = TRUE)

suf$hgrossIncome <- rowSums(suf[, c("hy040g", "hy050g", "hy060g",
    "hy070g", "hy080g", "hy090g", "hy110g")], na.rm = TRUE)
```

We now skip to list the code for adding further variables, since it should now be clear how to do this using functions `simCategorical` or `simContinuous`. In the end, we can produce all (approx.) 450 variables or more, if country specific variables should also be included. Important is the function argument `regModel` that should be used with care, i.e. a regression model can be specified for each variable to simulate. The better the model, the better the quality of the synthetic data set.

8.7.3 Optionally: Draw a Sample from the Population

In case one wants to work with a sample instead of the whole population, we can extract a sample from our simulated synthetic population using any complex sample design.

We choose to use stratified group sampling. We use the region variable (`db040`) for stratification. We select households and then all members of one specific household are presented in the synthetic sample. Consequently, the number of households in the synthetic PUF is the same as in the original microdata, but the number of individuals may differ.

```
## ----table of number of households per region (used drawing the puf)
x2 <- data.table(x[,c("db040","db030")])
setkey(x2, db030)
hh <- x2[, unique(db040), by = db030]
tab <- table(hh$V1)
## number of households in all districts
tab

   ##
   ## AT11 AT12 AT13 AT21 AT22 AT31 AT32 AT33 AT34
   ##  200 1140 1278  376  828 1034  386  492  243

tab <- as.numeric(table(hh$V1))
```

```
## number of households:
sum(tab)

   ## [1] 5977

## stratified group sampling, equal size
set.seed(23456)
dim(pop(eusilcP))

   ## [1] 8368615      40

class(pop(eusilcP))

   ## [1] "data.table" "data.frame"

puf <- draw(data.frame(pop(eusilcP)),
             design = "db040",
             grouping = "db030",
             size = tab)

## rename weight vector
colnames(puf)[which(colnames(puf) == ".weight")] <- "rb050"
puf$rb050 <- puf$rb050 * myfactor

## size of the data
dim(puf)

   ## [1] 13515    41

nrow(puf[!duplicated(puf$db030), ])

   ## [1] 5977
```

```
if(calib){
  totals1 <- tableWt(eusilc[, c("rb090","db040")], weights=eusilc$rb050)
  weights2 <- calibSample(puf, as.data.frame(totals1), w = puf$rb050)
  puf$rb050_calib <- weights2$final_weights
}
```

If your sampling weights differ a lot from your original data set, you did not specify the correct sampling design (country specific) and you may have to use another sampling design to draw from your population.

8.7.4 *Exploration of the Final Synthetic Population and Sample*

Basically, all utility measures described in Chap. 5 can be used to evaluate the quality of the data as well as risk measures defined in Sect. 6.3 can be used to estimate the disclosure risk. While the disclosure risk is very low (see Sect. 6.3 and Templ and

Alfons 2010), we should have a closer look at the data utility. First, let us extract the population and sample data from our object `eusilcP`. We then may estimate the number of individuals in the population (weighted sum, using the sampling weights of the raw survey data) and calculate the number of individuals in the synthetic population. This we also apply for the number of households (using package **data.table** for faster calculations). We can observe that these numbers are very similar between the synthetic population and sample.

```
pop <- pop(eusilcP)
sample <- samp(eusilcP)
## Estimated number of individuals in the population
sum(sample$rb050)

   ## [1] 8344294

## Calculated number of individuals in the synthetic population
nrow(pop)

   ## [1] 8344287

## Estimated number of households in the population
sum(sample[!duplicated(sample$db030), rb050])

   ## [1] 3674279

## Calculated number of households in the synthetic population
library("data.table")
DT <- data.table(pop)
setkeyv(DT, "db030")
nrow(unique(DT))

   ## [1] 3674279
```

For any variable or any combination of variables we can do some (cross-)tabulation and look at differences. For the estimated number of men and women in the population, we use the function `tableWt` (weighted counts). The number are almost identical to the one from the synthetic population.

```
# Distribution of individuals by gender
tableWt(sample$rb090, weights = sample$rb050)
```

```
## x
##    male  female
## 4084158 4260136
```

```
table(pop$rb090)
```

```
##
##    male  female
## 4083827 4260460
```

We show one additional tablulation, from where we can see excellent results for most categories except the categories with low counts.

```
## Distribution of individuals by education (highest ISCED level attained)
tabPop <- table(pop$pe040)
p1 <- tabPop
p1
```

```
##
##     0-1       2       3       4       5       6     NAs
##  117027 1610687 3377424  698016 1040716  186437 1313980
```

```
tabSample <- tableWt(sample$pe040, weights = sample$rb050)
p2 <- tabSample
p2
```

```
## x
##     0-1       2       3       4       5       6     NAs
##   65082 1638248 3418072  709181 1049552  151534 1312625
```

```
(p1 - p2)/p2 * 100
```

```
##
##        0-1          2          3          4          5
## 79.8146953 -1.6823460 -1.1892084 -1.5743513 -0.8418830
##          6        NAs
## 23.0331147  0.1032283
```

We stop to produce more tables but we like to mention that users should produce more tables and compare the results.

Another issue is to look at popular indicators. We select the Gini coefficient that is in the following estimated using the raw sample survey and calculated using the synthetic population. The results are similar.

```
## Gini coefficient for equivalized disposable income for income > 15000
gini("pgrossIncome", data = pop[pop$pgrossIncome > 0,])$value

   ## [1] 43.23417

gini("pgrossIncome", weights="rb050",
   data=sample[sample$pgrossIncome > 0,])$value

   ## [1] 42.50472
```

One can additionally look at variance of the Gini, further indicators but also results from regression modeling. In order to stay in the limits of pages, we skip this but we

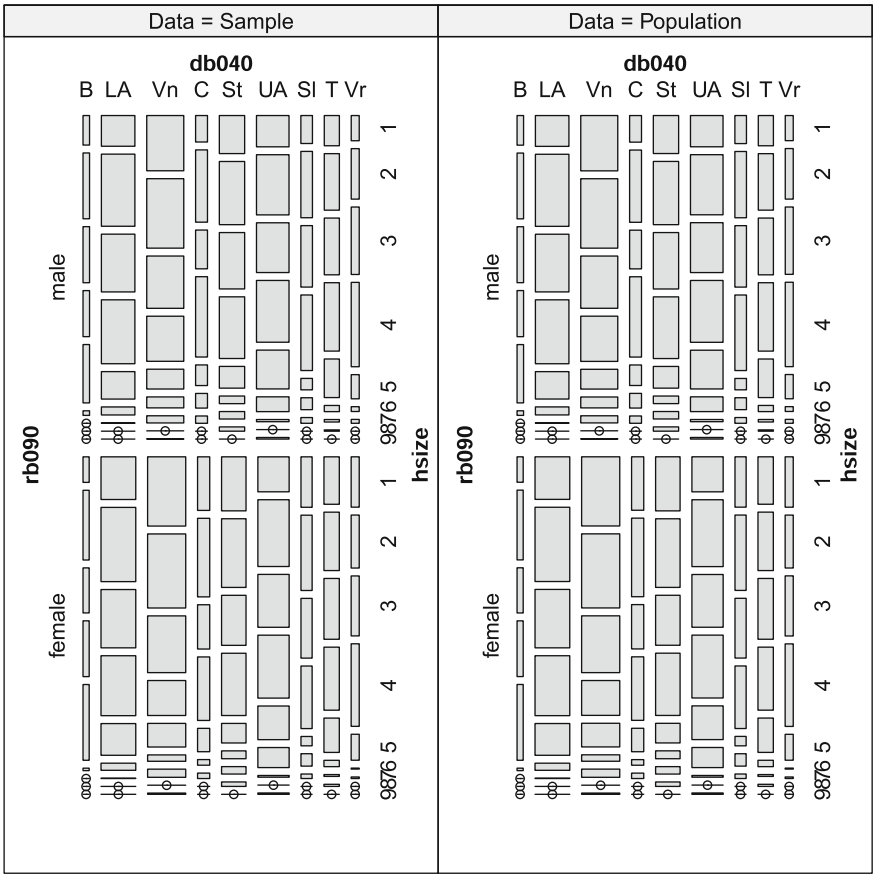


Fig. 8.9 Estimated frequencies on the combination of gender (`rb090`), region (`db040`) and `hsize` from the raw survey data (*left*) and calculated frequencies obtained from the synthetic population. Originally published in Templ et al. (2017). Published with kind permission of ©Matthias Templ, Bernhard Meindl, Alexander Kowarik and Oliver Dupriez 2017 under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium provided the original author(s) and source are credited

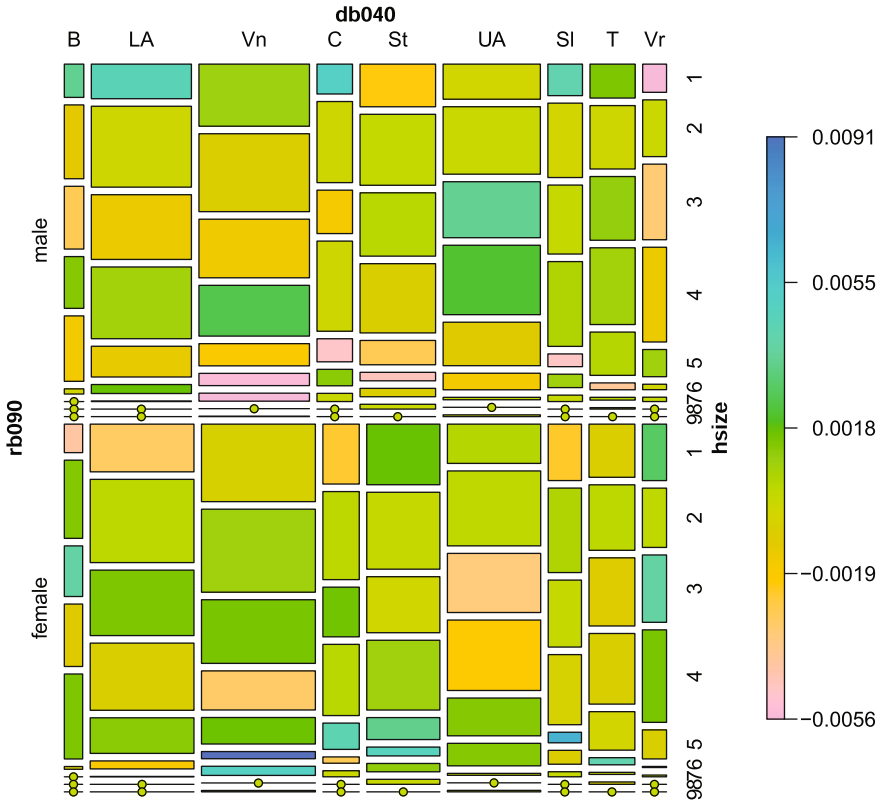


Fig. 8.10 Differences of estimated frequencies on the combination of gender (`rb090`), region (`db040`) and `hsize` from the raw survey data (*left*) and the calculated frequencies obtained from the synthetic population

want to show some nice exploratory visualization tools to compare the sample and the synthetic population.

For this purpose, we first evaluate the expected (i.e., estimated) and realized (i.e., simulated/synthetic) population sizes via mosaic plots (Hartigan and Kleiner 1981; Hofmann 2003). We use the function `spTable` to produce an object of the same class. A character string specifying the plot method in function `spMosaic`. Possible values are "split" to plot the expected population sizes on the left hand side and the realized population sizes on the right hand side, and "color" If method is "split", the two tables of expected and realized population sizes are combined into a single table, with an additional conditioning variable indicating expected and realized values. Let us have a look on the split method—the results are presented in Fig. 8.9. The results looks excellent and now differences can be detected.

The realized population sizes colored according to relative differences with expected population sizes is visualized in Fig. 8.10. D differences can be detected, e.g. for females in 6-person households in the region of Vienna (`Vn`). However, also

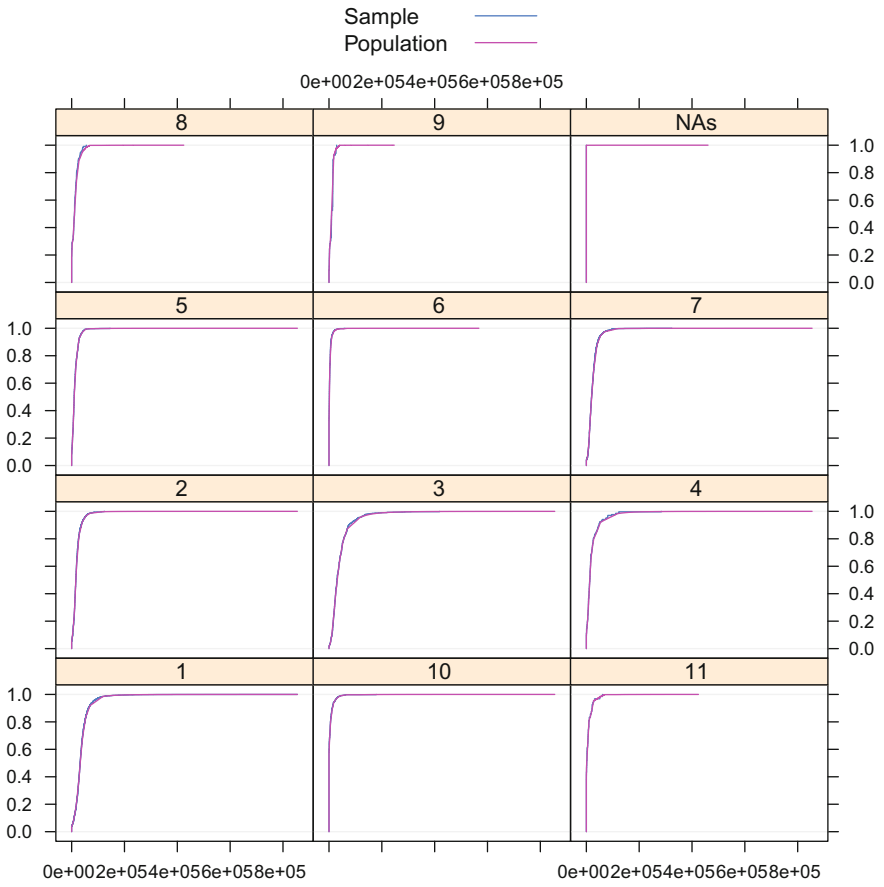


Fig. 8.11 Cumulative distribution function of variable `pgrossIncome` conditioned on educational level (`pl031`) estimated from the raw survey data and the synthetic population

this particular difference between the synthetic population and the raw sample survey is negligible.

For continuous variables we may plot cumulative distribution functions, possibly broken down according to conditioning variables and taking into account sample weights for the cumulative distribution functions of the sample. We simply can work with our object `eusilcP` of class *simPopObj* that contains the survey sample and synthetic population data. A character vector specifying the columns of data available in the sample and the population to be plotted, and an optional character vector (of length 1, if used) specifying the conditioning variable. Weights are directly extracted from the input object and are taken into account by adjusting the step height. To be precise, the weighted step height for an observation is defined as its weight divided by the sum of all weights. The result is produced with the following code and shown in Fig. 8.11. The (weighted) cumulative distribution functions of the gross income

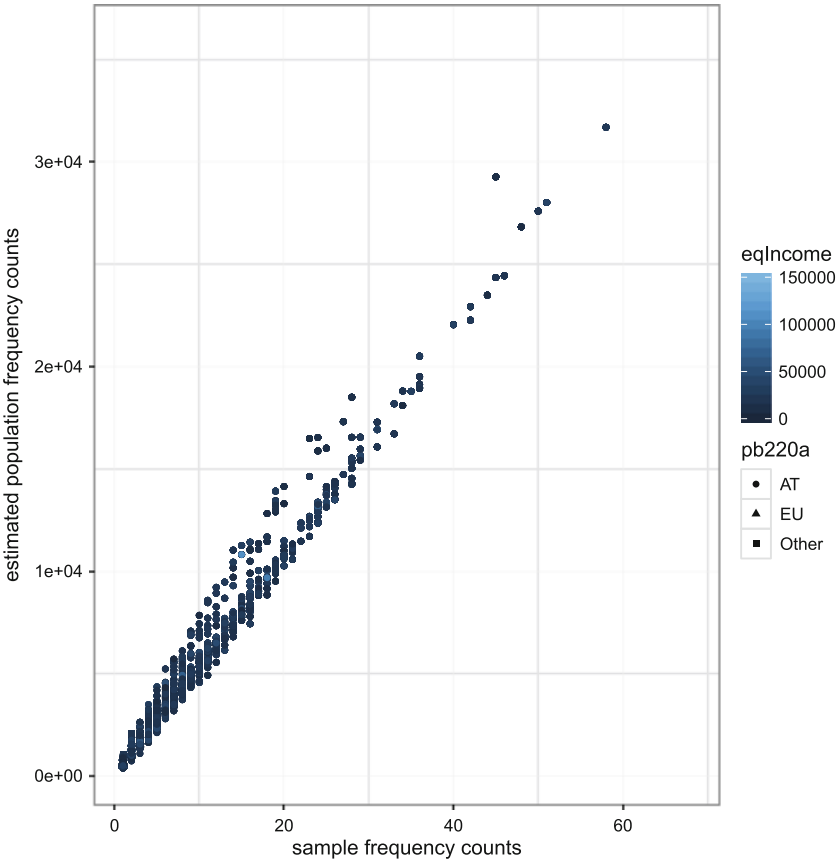


Fig. 8.12 Parallel boxplot of variable `pgrossIncome` conditioned on educational level (`pl031`) comparing the raw survey data and the synthetic population

from the raw survey data is very similar to the cumulative distribution functions calculated from the synthetic population.

We can also produce and carry out some comparisons with box-and-whisker plots of continuous or semi-continuous variables, possibly broken down according to conditioning variables and taking into account sample weights. Again we can plot an object of class *simPopObj* directly, containing survey sample and synthetic population data, and specifying a character vector for the columns of data available in the sample and the population to be plotted. Missing values are ignored for producing box plots and weights are directly extracted from the input object inp. The results are shown in Fig. 8.12. The boxplots based on the raw survey data are almost identical to the one from the synthetic population (again the gross income is in focus). Minor differences can be only seen for large values of `pl030`. The reason is that there exists only few large values in the raw survey sample having such large values, and thus

statistical uncertainty is larger for these categories. However, if we would produce more than one synthetic population, we know from the limit theorems in statistics that in average the results should be the same as for the raw survey data.

We showed a lot of comparisons between the raw survey data and the synthetic population. Even more comparisons are shown in Alfons et al. (2011), Templ and Filzmoser (2014) and the software package **simPop** is shown in more detail in Templ et al. (2017). A lot of additional results can be produced using other variables and methods to evaluate the quality of the data (see Chap. 5). Equivalent to the approach to compare a synthetic population with the raw survey data, synthetic survey data (as, for example, produced in Sect. 8.7.3) can be compared to the raw survey data. We do not show further results on this and note that, after reading Chap. 5 as well as this case study, the production of further results on data utility is now straightforward to do.

References

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. http://dx.doi.org/10.1007/s10260-011-0163-2.

Atkinson, T., Cantillon, B., Marlier, E., & Nolan, B. (2002). *Social indicators: The EU and social inclusion*. New York: Oxford University Press.

Beblot, M., Beniger, D., Heinze, A., & Laisney, F. (2003). *Methodological issues related to the analysis of gender gaps in employment, earnings and career progression*. European Commission Employment and Social Affairs DG: Final project report.

Belfield, R. (1999). *Pay inequalities and economic performance: A review of the UK literature*. Centre for Economic Performance London School of Economics Houghton Street, London: Technical Report PiEP Report.

Bowles, S., Gintis, H., & Osborne, M. (2001). The determinants of earnings: A behavioral approach. *Journal of Economic Literature*, *39*, 1137–1176.

Bruch, C., Münnich, R., & Zins S. (2011). Variance estimation for complex surveys. Research Project Report WP3—D3.1, FP7-SSH-2007-217322 AMELI. http://ameli.surveystatistics.net.

Caju, P., Fuss, C., & Wintr, L. (2009a). Understanding sectoral differences in downward real wage rigidity: Workforce composition, institutions, technology and competition. Working paper series no. 1006, European Central Bank. Wage dynamics network. http://www.ecb.int/pub/pdf/scpwps/ecbwp1006.pdf.

Caju, P., Rycx, F., & Tojerow, I. (2010). Wage structure effects of international trade: Evidence from a small open economy. Working paper series no. 1325, European Central Bank. Wage dynamics network. http://www.ecb.int/pub/pdf/scpwps/ecbwp1325.pdf.

Caju, P., Rycx, F., & Tojerow, I. (2009b). Inter-industry wage differentials: How much does rent sharing matter? *Journal of the European Economic Association*, *79*(4), 691–717.

Dell'Aringa, C., Ghinetti, P., & Lucifora, C. (2000). Pay inequality and economic performance in Italy: A review of the applied literature. In *Proceedings of the LSE Conference*, 3–4 November 2000, London.

Deville, J.-C., Särndal, C.-E., & Sautory, O. (1993). Generalized raking procedures in survey sampling. *Journal of the American Statistical Association*, *88*(423), 1013–1020.

Deville, J.-C., & Särndal, C.-E. (1992). Calibration estimators in survey sampling. *Journal of the American Statistical Association*, *87*(418), 376–382.

Dupray, A., Nohara, H., & Béret, P. (1999). *Pay inequality and economic performance: a review of the french literature*. Centre for Economic Performance London School of Economics, London: Technical Report PiEP Report.

Dybczak, K., & Galuscak, K. (2010). Changes in the Czech wage structure: Does immigration matter? Working paper series no. 1242, European Central Bank. Wage dynamics network. http://www.ecb.int/pub/pdf/scpwps/ecbwp1242.pdf.

Efron, R. G., & Tibshirani, R. G. (1993). *An introduction to the bootstrap*. New York: Chapman and Hall.

Eurostat (2004). *Description of target variables: Cross-sectional and longitudinal*. EU-SILC 065/04. Luxembourg: Eurostat.

Eurostat (2014a). Ses microdata for scientific purposes: How to obtain them? http://epp.eurostat.ec.europa.eu/portal/page/portal/microdata/documents/EN-SES-MICRODATA.pdf.

EU-SILC (2009). Algorithms to compute social inclusion indicators based on EU-SILC and adopted under the Open Method of Coordination (OMC). EU-SILC LC-ILC/39/09/EN-rev.1, Directorate F: Social and information society statistics Unit F-3: Living conditions and social protection. Luxembourg: EUROPEAN COMMISSION, EUROSTAT.

Eurostat (2014b). Anonymisation method for ses 2002 and 2006 microdata—synthesis. http://epp.eurostat.ec.europa.eu/portal/page/portal/microdata/documents/SES_anonymisation_method.pdf.

EU-SILC (2004). Common cross-sectional EU indicators based on EU-SILC; the gender pay gap. EU-SILC 131-rev/04, Working group on Statistics on Income and Living Conditions (EU-SILC). Luxembourg: Eurostat.

Franconi, L., & Polettini, S. (2004b). Individual risk estimation in μ-ARGUS: A review. In *Privacy in statistical databases*. Lecture Notes in Computer Science (pp. 262–272). Springer.

Frick, B., & Winkelmann, K. (1999). Pay inequalities and economic performance: A review in literature. Technical Report Research Report HPSE-CT-1999-00040, Ernst-Moritz-Arndt-Universität Greifswald, Greifswald.

Geissberger, T. (2009). *Verdienststrukturerhebung 2006*. Statistik Austria: Struktur und Verteilung der Verdienste in Österreich. ISBN 978-3-902587-97-8.

Geissberger, T. (2010). Frauenbericht. teil 4: Sozioökonomische studien. Technical Report 4, Federal Minister for Women and the Civil Service of Austria, Wien.

Geissberger, T., & Knittler, K. (2010). Niedriglöhne und atypische Beschäftigung in Österreich. *Statistische Nachrichten6*, 448–461.

Gini, C. (1912). Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche. *Studi Economico-Giuridici della R. Università di Cagliari3*, 3–159.

Groshen, E. (1991). The structure of the female/male wage differential. *Journal of Human Resources*, *26*, 455–472.

Hartigan, J. A., & Kleiner, B. (1981). Mosaics for contingency tables. In W. F. Eddy (Ed.), *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface* (pp. 268–273). New York: Springer.

Hofmann, H. (2003). Constructing and reading mosaicplots. *Computational Statistics & Data Analysis*, *43*(4), 565–580.

Ichim, D., & Franconi, L. (2007). Disclosure scenario and risk assessment: Structure of earnings survey. *In Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality, Manchester.*,. doi:10.2901/Eurostat.C2007.004.

Lorenz, M. O. (1905). Methods of measuring the concentration of wealth. *Publications of the American Statistical Association*, *9*(70), 209–219.

Marsden, D. (2010). Pay inequalities and economic performance. Technical Report PiEP Final Report V4, Centre for Economic Performance London School of Economics, London.

Messina, J., Izquierdo, M., Caju, P., Duarte, C. F., & Hanson, N. L. (2010). The incidence of nominal and real wage rigidity: An individual-based sectoral approach. *Journal of the European Economic Association*, *8*(2–3), 487–496.

Nolan, B., & Russel, H. (2001). *Pay inequality and economic performance in ireland: A review of the applied literature*. The Economic and Social Research Institute, Dublin: Technical Report PiEP Report.

Pointner, W., & Stiglbauer, A. (2010). Changes in the austrian structure of wages. Working paper series no. 1268, European Central Bank. Wage dynamics network. http://www.ecb.int/pub/pdf/scpwps/ecbwp1268.pdf.

Rinott, Y., & Shlomo, N. (2006). A generalized negative binomial smoothing model for sample disclosure risk estimation. In *Privacy in statistical databases* (pp. 82–93). Lecture Notes in Computer Science. Springer.

Simón, H. (2010). International differences in wage inequality: A new glance with european matched employer-employee data. *British Journal of Industrial Relations48*(2), 310–346.

Stephan, G., & Gerlach, K. (2005). Wage settlements and wage settings: Evidence from a multi-level model. *Applied Economics*, *37*, 2297–2306.

Templ, M., & Alfons, A. (2011). Variance estimation of social inclusion indicators using the R package **laeken**. Research Report CS-2011-3, Department of Statistics and Probability Theory, Vienna University of Technology. http://www.statistik.tuwien.ac.at/forschung/CS/CS-2011-3complete.pdf.

Templ, M. (2009). *New developments in statistical disclosure control and imputation: Robust statistics applied to official statistics*. Südwestdeutscher Verlag fuer Hochschulschriften.

Templ, M., & Alfons, A. (2010). Disclosure risk of synthetic population data with application in the case of EU-SILC. In *Privacy in Statistical Databases.* Lecture Notes in Computer Science (pp. 174–186). Springer. ISBN 978-3-642-15837-7.

Templ, M. (2011a). 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.

Templ, M. et al. (2011a). R packages plus manual. Research Project Report WP10–D10.3, FP7-SSH-2007-217322 AMELI. http://ameli.surveystatistics.net.

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.

Templ, M. (2015). Quality indicators for statistical disclosure methods: A case study on the structural earnings survey. *Journal of Official Statistics*, *13*(4), 737–761.

Templ, M., Meindl, B., & Kowarik, A. (2015). Statistical disclosure control for micro-data using the R package sdcMicro. *Journal of Statistical Software*, *67*(1), 1–37.

Templ, M., Meindl, B., Kowarik, A., & Dupriez, O. (2017). Simulation of synthetic complex data: The R-package simPop. *Journal of Statistical Software*, 1–38. Accepted for publication in December 2015.

Software Versions Used in the Book

All computations in this paper were performed using with the following R session:

- R version 3.3.2 (2016-10-13), `x86_64-apple-darwin13.4.0`
- Base packages: base, datasets, grDevices, graphics, grid, methods, stats, tcltk, tools, utils
- Other packages: MASS 7.3-40, RGtk2 2.20.31, VIM 4.1.0, boot 1.3-16, brew 1.0-6, cairoDevice 2.22, colorspace 1.2-6, data.table 1.9.4, devtools 1.8.0, foreign 0.8-63, gWidgets 0.0-54, gWidgetsRGtk2 0.0-83, ggplot2 1.0.1, knitr 1.10.5, laeken 0.4.6, lattice 0.20-31, rmarkdown 0.7, robCompositions 2.0.1, robustbase 0.92-3, sdcMicro 5.0.0, sets 1.0-14, simPop 0.3.0, stringr 1.0.0, vcd 1.3-2, xtable 1.7-4

All results are fully reproducible since they are produced *on the fly*, i.e. any table, figure, calculation or output included in this book is produced from "one" text file by using **knitr** (Xie 2013, 2014a, b).

References

Xie, Y. (2013). *Dynamic documents with R and knitr*. Boca Raton: Chapman and Hall/CRC. ISBN 978-1482203530. http://yihui.name/knitr/.

Xie, Y. (2014a). *knitr: A general-purpose package for dynamic report generation in R*. R package version 1.6. http://yihui.name/knitr/.

Xie, Y. (2014b). knitr: A comprehensive tool for reproducible research in R. In V. Stodden, F. Leisch, & R. D. Peng (Eds.), *Implementing reproducible computational research*. Boca Raton: Chapman and Hall/CRC. ISBN 978-1466561595. http://www.crcpress.com/product/isbn/9781466561595.

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4

Solutions

1.1 To look at the help index of package **sdcMicro** (assuming the package is already installed, use:

```
help(package = "sdcMicro")
```

To execute examples of a help file, e.g. of the help file of `?globalRecode`, use

```
library("sdcMicro")
example("globalRecode")
```

1.2 We load the `testdata` and show the commands to receive the required information.

```
data(testdata, package = "sdcMicro")
# str(testdata)
# head(testdata)
# colnames(testdata)
# ?testdata
```

And we load the data set `eusilc`.

```
library("laeken")
data("eusilc")
```

1.3 We run a part of `globalRecode` (example section)

```
data("testdata2")
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- globalRecode(sdc, column="urbrur", breaks=5)
```

and we list the names of all slots

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4

```
##  [1] "origData"          "keyVars"
##  [3] "pramVars"          "numVars"
##  [5] "ghostVars"         "weightVar"
##  [7] "hhId"              "strataVar"
##  [9] "sensibleVar"       "manipKeyVars"
## [11] "manipPramVars"     "manipNumVars"
## [13] "manipGhostVars"    "manipStrataVar"
## [15] "originalRisk"      "risk"
## [17] "utility"           "pram"
## [19] "localSuppression"  "options"
## [21] "additionalResults" "set"
## [23] "prev"              "deletedVars"
```

We access the slot risk either with `slot` or `?get.sdcMicroObj`

```
risk <- slot(sdc, "risk")
str(risk)
```

```
## List of 3
##  $ global    :List of 5
##   ..$ risk     : num 0.0465
##   ..$ risk_ER  : num 4.33
##   ..$ risk_pct : num 4.65
##   ..$ threshold: num 0
##   ..$ max_risk : num 0.01
##  $ individual: num [1:93, 1:3] 0.0465 0.0465 0.0465 0.0465
    0.0465 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "risk" "fk" "Fk"
##  $ numeric   : num 1
```

```
head(risk$individual[, "risk"])
```

```
## [1] 0.04651687 0.04651687 0.04651687 0.04651687 0.04651687
## [6] 0.04651687
```

2.1 First, the data set *eusilc* should be loaded into R. If the package **laeken** is not installed, the first lines of the following code will install the package. Afterwards the structure of the eusilc data is explored.

```
if (!require(laeken)){
        install.packages(laeken)
}
library(laeken)
data(eusilc)
str(eusilc)
```

```
## 'data.frame': 14827 obs. of 28 variables:
## $ db030 : int 1 1 1 2 2 2 2 3 4 4 ...
## $ hsize : int 3 3 3 4 4 4 4 1 5 5 ...
## $ db040 : Factor w/ 9 levels "Burgenland","Carinthia",..: 6 6 6 6 6 6
6 8 8 8 ...
## $ rb030 : int 101 102 103 201 202 203 204 301 401 402
...
## $ age : int 34 39 2 38 43 11 9 26 47 28 ...
## $ rb090 : Factor w/ 2 levels "male","female": 2 1 1 2 1 1 1 2 1 1 ...
## $ pl030 : Factor w/ 7 levels "1","2","3","4",..: 2 1 NA 7 1 NA NA 7 3 1 ...
## $ pb220a : Factor w/ 3 levels "AT","EU","Other": 1 3 NA 1 1
NA NA 1 3 1 ...
## $ py010n : num 9756 12472 NA 12487 42821 ...
## $ py050n : num 0 0 NA 0 0 ...
## $ py090n : num 0 0 NA 0 0 ...
## $ py100n : num 0 0 NA 0 0 NA NA 0 0 0 ...
## $ py110n : num 0 0 NA 0 0 NA NA 0 0 0 ...
## $ py120n : num 0 0 NA 0 0 NA NA 0 0 0 ...
## $ py130n : num 0 0 NA 0 0 NA NA 0 0 0 ...
## $ py140n : num 0 0 NA 0 0 NA NA 0 0 0 ...
## $ hy040n : num 4274 4274 4274 0 0 ...
## $ hy050n : num 2428 2428 2428 1550 1550 ...
## $ hy070n : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy080n : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy090n : num 33.39 33.39 33.39 2.13 2.13 ...
## $ hy110n : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy130n : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy145n : num 0 0 0 0 0 ...
## $ eqSS : num 1.8 1.8 1.8 2.1 2.1 2.1 2.1 1 2.8 2.8  ...
## $ eqIncome: num 16091 16091 16091 27076 27076 ...
## $ db090 : num 505 505 505 493 493 ...
## $ rb050 : num 505 505 505 493 493 ...
```

The help file gives us information on the variables; type `?eusilc` in R. It reports that this data set consists of persons of about 6000 households. All persons in a selected household are sampled.

(a) We cannot detect any direct identifiers such as social security number, names or addresses of individuals.
(b) Categorical key variables, variables which might be available in other public data bases, might be age, gender (*rb090*), region (*db040*), economic status (*pl030*) and citizenship (*pb220a*).
(c) In any case, some of the income components could be available in public data bases. The exact values can be used to link information.
(d) Income is sensible information. It could be used, for example, for insurance companies or credit institutions to evaluate person's financial standing.

2.2 The help file (`?ses`) gives us information on the variables. It reports that this data set consists of 115,691 employees of various enterprises.

(a) We cannot detect any direct identifiers such as social security number, names or addresses of individuals.
(b) Categorical key variables, variables which might be available in other public data bases, might be age, sex, location, NACE1, size, education and occupation. We note that we have two scenarios, one for disclosing information on enterprises (key variables: location, NACE1, size) and one on employees (all previously mentioned key variables). If we want to reach k-anonymity, we have to do it for the enterprises on enterprise level (and corresponding key variables), then transfer the suppression pattern to employment level, and then ensure k-anonymity for employees.
(c) Continuous key variables might be earnings and earnings per hour.
(d) Earnings is sensible information. It could be used, for example, for insurance companies or credit institutions to evaluate person's financial standing.

2.3 The re-identification risk of observation 3 and 8 might be moderate. First, we can look for sample uniqueness.

```
ckeys <- c("db040", "age", "rb090", "pl030")
fr <- freqCalc(eusilc, keyVars=ckeys)
fr$fk[8] == 1

  ## [1] TRUE
```

(For details on function `freqCalc`, see `?freqCalc` and Sect. 3.5.1).
However, the sampling weight of observation 8 is

```
eusilc[8, "rb050"]

  ## [1] 868.2204
```

Now, we can expect that many persons may exist with similar characteristics (this depends on the survey sampling design). The risk might be low but one can even try to lower it for observations being sample unique.

The combination of categorical key variables of observation 3

```
eusilc[3, ckeys]

  ##   db040 age rb090 pl030
  ## 3 Tyrol   2  male  <NA>
```

occurs 15 times in the sample:

```
fr$fk[3]

  ## [1] 15
```

The disclosure risk is without doubt very low since already 15 individuals with the same characteristics are present in the sample.

Note that *eusilc* consists of persons in households. Therefore, the risk estimation has to consider the household structure of the data. See Sect. 3.6 for further reading.

```
require("laeken")
data(eusilc)
## closer look at observation 8
key <- c("db040","age","rb090","pl030")
eusilc[8, key]
```

```
##    db040 age  rb090 pl030
## 8 Vienna  26 female     7
```

```
## to count how often this key is present in the sample
## you can use functions aggregate(), by() or even
## the new dplyr package for faster computations:
DT <- data.table(eusilc[,key])
DT[, count:=.N, by = names(DT)]
```

```
##                db040 age  rb090 pl030 count
##     1:         Tyrol  34 female     2     3
##     2:         Tyrol  39   male     1     8
##     3:         Tyrol   2   male    NA    15
##     4:         Tyrol  38 female     7     5
##     5:         Tyrol  43   male     1    20
##    ---
## 14823: Lower Austria  16 female     4    10
## 14824: Upper Austria  38 female     1     6
## 14825:         Tyrol  31   male     1     5
## 14826:         Tyrol  60   male     1     3
## 14827:         Tyrol  53 female     6     1
```

```
DT[8,]
```

```
##     db040 age  rb090 pl030 count
## 1: Vienna  26 female     7     1
```

```
DT[3,]
```

```
##    db040 age rb090 pl030 count
## 1: Tyrol   2  male    NA    15
```

You see that observation 8 is unique in the data set while observation 3 has the same entries as the other 14 observations. The probability of re-identifying observation 8 is therefore surely higher than for observation 3. However, since we deal with a complex survey sample including information on the finite population in terms of the sampling weights, additional effort is necessary to estimate the disclosure risk. See Sect. 3 for more details.

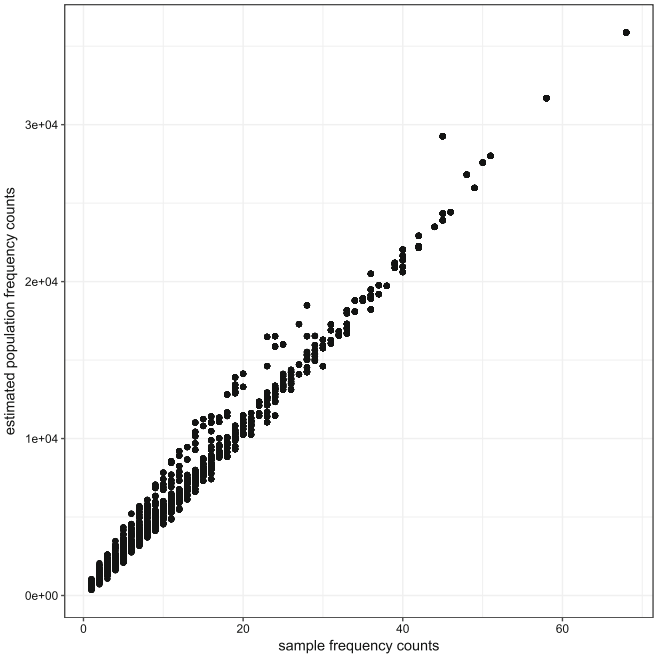
2.4 (a) Looking at Fig. 2.2, the microaggregation method (lower curve) clearly outperforms adding noise methods. Both the disclosure risk and the information loss are always lower. (b) Statisticians/experts on disclosure risk should propose a threshold of risk around 0.09 (that might be accepted by the management of the data holder), because microaggregation can then be used with much lower information loss. If they do not agree, you have to add additive noise, accepting information loss around 0.7. (c) When the risk threshold is 0.2 and the maximum information loss is 1.5, you can basically use any of the three methods, but microaggregation outperforms adding additive noise using this data set.

3.1 We take the `eusilc` data set from package **laeken** with `age`, `pb220a` (citizenship), `pl030` (education level), `rb090` (gender) and `hsize` (household size) as categorical key variables. First, we create an object of class *sdcMicroObj*.

```
library("laeken")
data(eusilc)
sdc <- createSdcObj(eusilc,
                    keyVars = c("age", "pb220a", "pl030",
                                "rb090", "hsize"),
                    weightVar = "rb050")
```

We access the frequency counts and plot them.

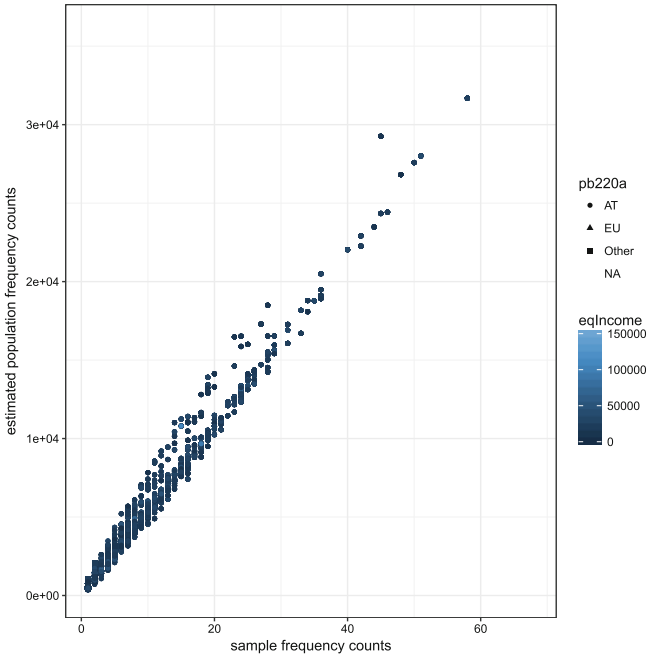
```
risk <- get.sdcMicroObj(sdc, "risk")$individual
freq <- data.frame(risk[, c("fk", "Fk")])

library("ggplot2")
gg <- ggplot(freq, aes(x=fk, y=Fk)) +
        geom_point() +
        xlab("sample frequency counts") +
        ylab("estimated population frequency counts") +
        theme_bw()
print(gg)
```



We see that there is a linear trend: the higher the sample frequencies, the higher the population frequencies.

We can further investigate to see if some groups have lower frequencies. Also, we can observe the relation with income.

```
eusilc$fk <- freq$fk
eusilc$Fk <- freq$Fk
library("ggplot2")
gg <- ggplot(eusilc, aes(x=fk, y=Fk, shape=pb220a,
                         colour=eqIncome)) +
              geom_point() +  xlab("sample frequency counts") +
              ylab("estimated population frequency counts") +
              theme_bw()
print(gg)
```



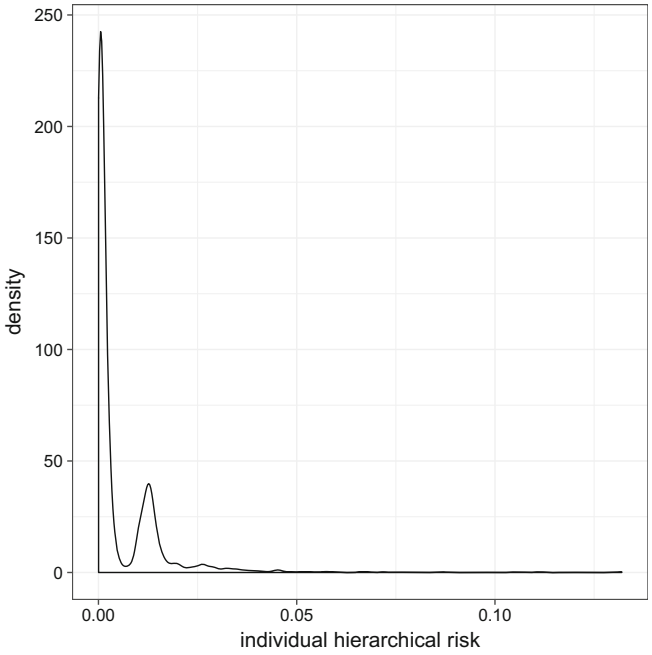
We see that individuals with higher incomes generally have lower frequencies according to the selected key variables.

3.2 We take the `eusilc` data set from package **laeken** with `age`, `pb220a` (citizenship), `pl030` (education level), `rb090` (gender) and `hsize` (household size) as categorical key variables. First, we create an object of class *sdcMicroObj*. We also specify the variable holding the information on weights as well as the household identifying variable.

```
library("laeken")
data(eusilc)
sdc <- createSdcObj(eusilc,
                    keyVars = c("age", "pb220a", "pl030",
                                "rb090", "hsize"),
                    weightVar = "rb050",
                    hhId = "db030")
```

We access the hierarchical risk and plot the distribution of the risk.

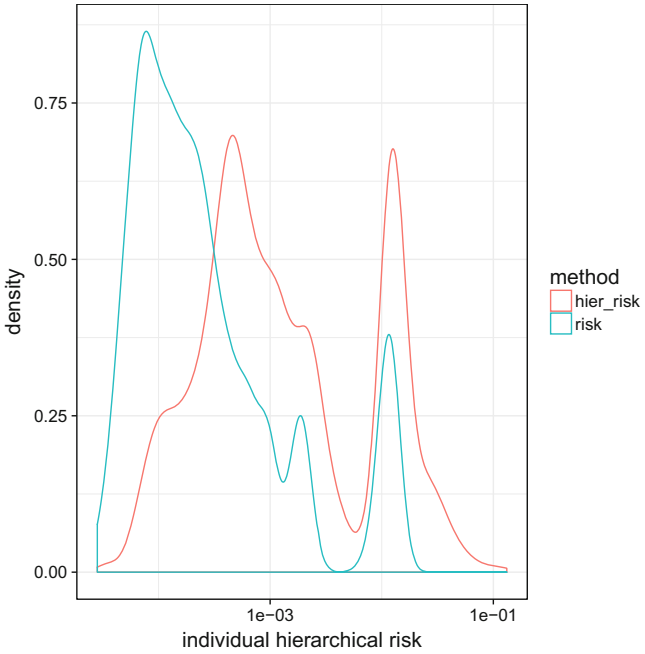
```
risk <- get.sdcMicroObj(sdc, "risk")$individual
eusilc$risk <- risk[, "risk"]
eusilc$hier_risk <- risk[, "hier_risk"]
r <- data.frame("risk" = c(risk[, "risk"], risk[, "hier_risk"]),
                "method" = rep(c("risk", "hier_risk"), each
                = nrow(risk)))
library("ggplot2")
gg <- ggplot(r[r$method == "hier_risk", ], aes(x = risk)) +
  xlab("individual hierarchical risk") +
  ylab("density") + geom_density()
print(gg)
```



The individual risks are rather low, but the risk should be reduced for few individuals, e.g. those which are greater than 0.05.

3.3 We better use a log-scale to compare the individual risk with the household risk.

```
library("ggplot2")
gg <- ggplot(r, aes(x = risk, group = method)) +
  geom_density(aes(colour=method)) +
  xlab("individual hierarchical risk") +
  ylab("density") + scale_x_log10()
print(gg)
```



We can observe that the hierarchical risks are (naturally) larger than the individual risks.

3.4 We make life easy and just take the first 500 observations of data set `eusilc` rather than sampling households.

```
data(eusilc)
keyVars <- c("db040", "hsize", "rb090", "age", "pb220a", "pl030")
eusilc$rb050_2 <- eusilc$rb050 * nrow(eusilc) / 500
sdc1 <- createSdcObj(eusilc, keyVars = keyVars,
weightVar = "rb050", hhId ="db030")
sdc2 <- createSdcObj(eusilc[1:500,],
keyVars = keyVars, weightVar =
"rb050_2", hhId = "db030")
form <- as.formula(paste(" ~ ", "db040 + hsize + rb090 +
age + pb220a + age:rb090 + age:hsize +
hsize:rb090"))
standardLLM <- as.formula(paste(c("fk",
as.character(form)),
collapse = ""))
m1 <- modRisk(sdc1, formulaM = form, bound = 5)@risk$model[1:2]
m1

   ## $gr1
   ## [1] 0.2964092
   ##
   ## $gr2
```

```
   ## [1] 0.3604305

m2 <- modRisk(sdc2, formulaM = form, bound = 5)@risk$model[1:2]
m2

   ## $gr1
   ## [1] 0.9110378
   ##
   ## $gr2
   ## [1] 0.934479
```

We can observe that the model-based global risk measure increases for smaller samples. This is problematic since a smaller sample should not result in higher risks. However, the number of predictors and interactions are already quite large compared to the number of observations so that the model results are no longer trustable. Estimating the global risk with log-linear models on such small data sets is no longer reliable.

3.5 If we look at the global risk estimated by summing up the household risks, we see that the global risk decreases when (further) sampling is applied.

```
print(sdc1, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main
   ## part of the data: 0
   ## Expected number of re-identifications: 57.49 (0.39 %)
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications: 199.16 (1.34 %)
   ## ------------------------------------------------------
        -----------------

print(sdc2, "risk")

   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main
   ## part of the data: 0
   ## Expected number of re-identifications: 0.29 (0.06 %)
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications: 0.93 (0.19 %)
   ## ------------------------------------------------------
        -------------------
```

3.6 Let us create our *sdcMicroObj* object first. Automatically, the risk will be re-estimated and stored in our *sdcMicroObj* object as soon as we apply `addNoise`.

```
data(testdata, package = "sdcMicro")
sdc <- createSdcObj(testdata,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- addNoise(sdc, method = "additive")
get.sdcMicroObj(sdc, "risk")$numeric

   ## [1] 0.008078603

sdc <- undolast(sdc)
sdc <- addNoise(sdc, method = "correlated2")
get.sdcMicroObj(sdc, "risk")$numeric

   ## [1] 0.3218341
```

The risk is lower for additive noise but we will see afterwards that the data utility is very low whenever this method is applied.

4.1 Let us create our *sdcMicroObj* object first.

```
library("laeken")
data("eusilc")
eusilc$hsize <- factor(eusilc$hsize)
sdc <- createSdcObj(eusilc,
                    keyVars = c("age", "pb220a", "pl030",
                                "rb090", "hsize"),
                    weightVar = "rb050",
                    hhId = "db030")
```

We take a look on the disclosure risk.

```
print(sdc, "risk")


   ## Risk measures:
   ##
   ## Number of observations with higher risk than the main
   ## part of the data: 0
   ## Expected number of re-identifications: 20.94 (0.14 %)
   ##
   ## Information on hierarchical risk:
   ## Expected number of re-identifications: 78.59 (0.53 %)
   ## ------------------------------------------------------------
   -----------
```

The number of expected re-identifications should be reduced through global recoding. We see that there are only few households with 8 or 9 inhabitants. Thus, one suggestion is to group them with 7-person households to a new group called `7-9`.

```
table(eusilc$hsize)

  ##
  ##    1    2    3    4    5    6    7    8    9
  ## 1745 3624 3147 3508 1815  630  252   88   18

sdc <- groupAndRename(sdc, "hsize", before = 1:9, after =
                          c(1:7, 7, 7))
## check
table(extractManipData(sdc)$hsize)

  ##
  ##     1
  ## 14827
```

It is always a good idea to look at the frequencies (`table()`) of all key variables. If we also do this for all key variables, we will find out that the variable age is given in years and grouping it into, for example, 5-year intervals will reduce the risk considerably. In addition, we form on group for all people older than 80.

```
labs <- c("0-4","5-9","10-14","15-19","20-24","25-29","30-34","35-39",
  "40-44","45-49","50-54","55-59","60-64","65-69","70-74","75-79","80+")
sdc <- globalRecode(sdc, column = "age",
                    breaks = c(seq(-1,80,5), 120),
                    labels=labs)
```

Now, the risk is reduced considerably without having recoded the data in great extent. More on evaluation of data utility, see Chap. 5 and related solutions of this chapter.

```
print(sdc, "risk")

  ## Risk measures:
  ##
  ## Number of observations with higher risk than the main
       part of the data:
  ##   in modified data: 0
  ##   in original data: 0
  ## Expected number of re-identifications:
  ##   in modified data: 0.18 (0.00 %)
  ##   in original data: 20.94 (0.14 %)
  ##
  ## Information on hierarchical risk:
  ## Expected number of re-identifications:
```

```
##   in modified data: 0.56 (0.00 %)
##   in original data: 78.59 (0.53 %)
## ------------------------------------------------------------
----------
```

4.2 Again, we use the `eusilc` data set from package **laeken** and the *sdcMicroObj* object produced in the previous example including all already done recodings. We ensure k-anonymity by

```
sdc <- kAnon(sdc)
sdc
```

```
## Data set with 14827 rows and 33 columns.
##  --> Categorical key variables: age, pb220a, pl030, rb090, hsize
##  --> Weight variable: rb050
## ----------------------------------------------------------------------
##
## Information on categorical Key-Variables:
##
## Reported is the number, mean size and size of the smallest category
## for recoded variables.
## In parenthesis, the same statistics are shown for the unmodified data.
## Note: NA (missings) are counted as seperate categories!
##
##  Key Variable Number of categories      Mean size
##           age                  18 (99)   866.235  (149.768)
##        pb220a                   4  (4)  4035.667 (4035.667)
##         pl030                   8  (8)  1729.000 (1729.571)
##         rb090                   2  (2)  7413.500 (7413.500)
##         hsize                   7  (9)  2118.143 (1647.444)
##  Size of smallest
##               464    (1)
##               283  (283)
##               177  (178)
##              7267 (7267)
##               358   (18)
## ----------------------------------------------------------------------
##
## Infos on 2/3-Anonymity:
##
## Number of observations violating
##  - 2-anonymity: 0 (original data: 1422)
##  - 3-anonymity: 98 (original data: 2364)
##
## Percentage of observations violating
##  - 2-anonymity: 0.000 % (original data: 9.591 %)
##  - 3-anonymity: 0.661 % (original data: 15.944 %)
##
---------------------------------------------------------------------
##
## Local Suppression:
##  KeyVar | Suppressions (#) | Suppressions (%)
##     age |               37 |            0.250
##  pb220a |                0 |            0.000
##   pl030 |                4 |            0.027
```

```
##   rb090 |               0 |           0.000
##   hsize |               0 |           0.000
## ----------------------------------------------------------------------
```

4.3 We use the `EIA` data and create an object of class *sdcMicroObj*.

```
data("EIA")
sdc <- createSdcObj(EIA,
                    keyVars = c("UTILNAME","STATE"),
                    numVars = c("RESSALES", "COMSALES",
                                "INDSALES", "OTHRSALES"))
sdc <- addNoise(sdc, method = "correlated2")
slot(sdc, "risk")$numeric

  ## [1] 0.1632454

sdc <- undolast(sdc)
sdc <- addNoise(sdc,
method = "additive", noise = 1) slot(sdc, "risk")$numeric


  ## [1] 1
```

In general, the additive noise method provides better protection; the disclosure risk is almost zero, while the risk for the correlated noise method is somewhere between [0, 16, 47%. However, we will see afterwards that the data utility is very low for the additive noise method. Moreover, we will motivate that this method should not be used since it simply destroys the structure of the data → this is the reason why the disclosure risk is so low.

4.4 We undo adding noise and apply microaggregation where we define the strata.

```
sdc <- undolast(sdc)
sdc <- microaggregation(sdc, strata_variables = "STATE")
```

4.5 Maybe not. Successfully attacking the categorical key variables is already difficult since 3-anonymity is given. However, consider the case where you have an external data base and you can link to, say four individuals with the same entries in the categorical key variables, and assume that only one continuous key variable has high (microaggregated) values. Then, you might be able to re-identify an individual by combining your information on continuous and categorical key variables. Even if you would apply microaggregation with a high number of the aggregation level, this might not help. This is indeed a dilemma since we learned how to anonymize categorical key variables and how to anonymize continuous key variables independently. If your data may allow re-identification in such a manner as reported here, microaggregation might be used with care; alternative methods might be used for the anonymization of continuous key variables.

5.1 We create our *sdcMicroObj* object as usual.

```
library("laeken")
data("eusilc")
eusilc$hsize <- factor(eusilc$hsize)
sdc <- createSdcObj(eusilc,
                    keyVars = c("age", "pb220a", "pl030",
                                "rb090", "hsize"),
                    numVars = c("eqIncome"),
                    weightVar = "rb050",
                    hhId = "db030")
```

Then we apply some recoding and local suppression.

```
labs <- c("1-9","10-19","20-29","30-39",
          "40-49","50-59","60-69","70-79","80-130")
sdc <- globalRecode(sdc, column = "age",
                    breaks = c(0,9,19,29,39,49,59,69,79,130),
                    labels = labs)
sdc <- groupAndRename(sdc, var = "pl030",
                 before = levels(eusilc$pl030),
                 after = c(1:5, "6-7", "6-7"))
sdc <- groupAndRename(sdc, var = "hsize",
                 before = levels(eusilc$hsize),
                 after = c(1:6, rep("7-9", 3)))
sdc <- kAnon(sdc)
sdc <- microaggregation(sdc)
```

Now do the cross tabulation.

```
library("simPop")
## raw survey data
tableWt(eusilc[, c("hsize", "pl030")], weights = eusilc$rb050)
```

```
##       pl030
## hsize      1      2      3      4      5      6      7
##     1 426561  76187  63050  28343 568613  22262  30646
##     2 639951 153905  70021  51922 829922  25064 169281
##     3 686578 152243  64877  92811 227699  14887 162354
##     4 662556 176243  55821 110242  88245  20738 156523
##     5 302249  58663  36502  84156  42261   4724  73238
##     6  82636  12908   3895  18998  32417   5194  36763
##     7  35848   5973   7704   8190  13081    543  10083
##     8  31868      0   1383    692   3859  10754      0
##     9   1621      0      0    475    857    764   1425
```

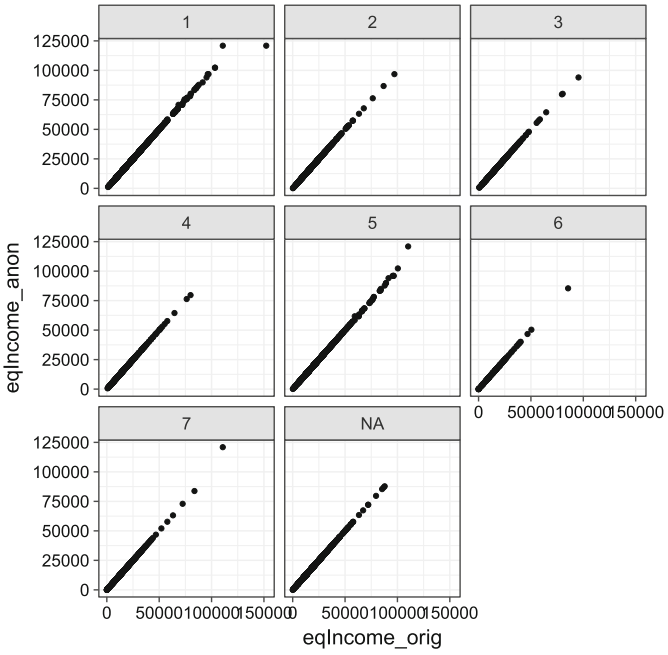
```
## modified data
eusilc_anon <- extractManipData(sdc)
tableWt(eusilc_anon[, c("hsize", "pl030")], weights = eusilc$rb050)
```

```
##       pl030
## hsize       1
##     1 6757264
```

We only see changes regarding combined categories.

5.2 We see minor changes for the anonymized variable `eqIncome`.

```
df <- data.frame("pl030" = eusilc$pl030,
                 "eqIncome_orig" = eusilc$eqIncome,
                 "eqIncome_anon" = eusilc_anon$eqIncome)
library("ggplot2")
ggplot(df, aes(x = eqIncome_orig, y = eqIncome_anon)) +
  geom_point() + facet_wrap(~pl030)
```



5.3 We apply a different anonymization method and compare the Gini coefficients to the results presented in Sect. 5.5.2.

```
sdc <- undolast(sdc)
sdc <- addNoise(sdc, method = "correlated2")
sdc@additionalResults$gini <- gini(inc = "eqIncome",
  weigths = "rb050",
  breakdown = "db040",
  data = extractManipData(sdc))$valueByStratum$value
## results from original data:
res <- gini(inc = "eqIncome",
            weigths = "rb050",
            breakdown = "db040",
            data = eusilc)$valueByStratum$value
## relative error (in percentages)
100*abs((res - sdc@additionalResults$gini)/res)
```

```
## [1] 0.30877474 0.73566728 0.09100412 0.85719784 0.15765447
## [6] 0.51683786 0.11343151 0.22932946 0.25908586
```

We would also see that the results on the Gini differences are similar to microaggregation.

6.3 Clearly, for (a) the researcher cannot rely on synthetic data since the results may differ depending on their detailed analysis. A synthetic data set can thus only be used in a remote execution setting; otherwise the data set must be anonymized using traditional methods. Since the disclosure risk might still not be very low, the researcher should agree on further contracts, which may permits him to publish further disclosive results.

However, for (b) the ideal data set would be a synthetic population. Remember, if needed, we can easily draw a sample from a population. The quality of the synthetic data is reasonably high and the data can be used to compare methods. This is often done, e.g. within the research projects mentioned in the foreword of the book.

6.4 Synthetic data are an ideal candidate for lecturing purposes because they can be very detailed and the disclosure risk is very low; while for anonymized data using traditional methods, a lot of categories must be combined (global recoding), values must be suppressed and typically (if no rigorous recodings and perturbations are applied) the disclosure risk is too high to give the data to the public for teaching purposes.

6.5 Simulate a population

We produce a synthetic population from the data set `testdata`. We mainly use the default values of the functions, but there generally is a lot of freedom in choosing other methods, parameters, etc.

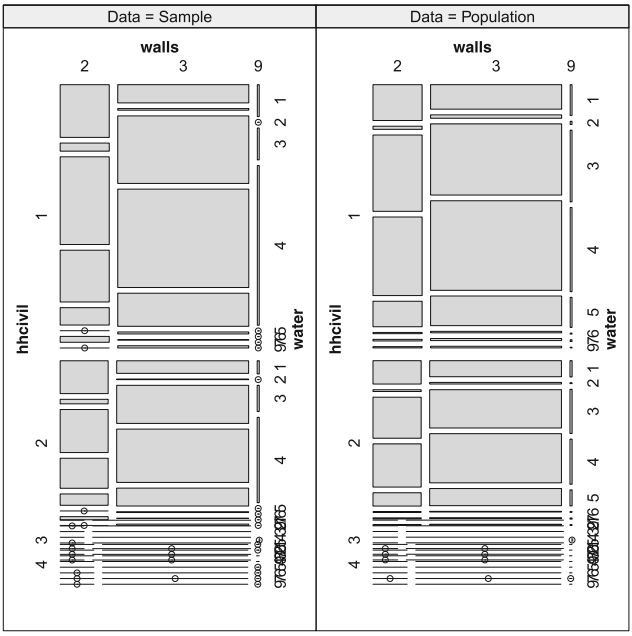
```
data(testdata, package = "sdcMicro")
pop <- specifyInput(testdata,
            hhid = "ori_hid",
            weight = "sampling_weight",
            strata = "urbrur")
## structure
pop <- simStructure(pop, basicHHvars = c("age","sex","hhcivil"))
## add categorical variables
pop <- simCategorical(pop,
          additional = c("roof","walls","water","electcon","relat"))
## add continuous variables
pop <- simContinuous(pop, additional = "expend")
pop <- simContinuous(pop, additional = "income")
pop <- simContinuous(pop, additional = "savings")
pop
```

```
##
## --------------
## synthetic population  of size
##  458000 x 19
##
## build from a sample of size
## 4580 x 16
```

```
## --------------
##
## variables in the population:
## ori_hid,hhsize,urbrur,age,sex,hhcivil,pid,weight,roof,walls,water,
## electcon,relat,expendCat,expend,incomeCat,income,savingsCat,savings
```

6.6 We produce one diagnostic plot to compare the synthetic population with the `testdata`.

```
tabp <- spTable(pop, select=c("hhcivil", "walls", "water"))
spMosaic(tabp)
```



7.1 Which variables do you consider as key variables for EU-SILC?

This question cannot be answered in a general manner since it depends on country-specific laws on privacy. However, if we assume a moderate scenario and that age, citizenship, educational level, gender and household size is available on other data bases, we should at least take these variables as categorical key variables.

Other variables such as "do you have a TV" cannot be seen as critical when a neighborhood scenario will not be taken into account (=the neighbor might know this information), because this variable will not be present in any other data set. Neighborhood scenarios are usually not taken into account. If some other variables should also be selected as categorical key variables should be discussed in detail with the subject matter specialists. The main question remains: is a variable also present in other data bases (or web) to which intruders might have access?

7.2 An alternative to produce a safe file for EU-SILC, especially when data with very low risk of disclosure should be produced, is to use the synthetic data simulation methods from Chap. 6.

7.3 EU-LFS (the EU Labor Force Survey) is a cross-sectional and longitudinal sample survey, coordinated by Eurostat and collected in EU member states (MS), some EFTA countries and some EU-candidate countries. The database comprises observations on labor market participation and persons outside the labor force. 13 variables might be defined as identifying variables: degree of urbanization, sex, age, nationality, occupation code, years of residence, highest level of education, country of birth, NACE, professional status, country of work, ILO working status and total number of persons in the household. See online resources on LFS for more information on these variables and the output of a project SGA on PUF (cab be easily found via any web search engine).

7.4 When speaking about an acceptable level of disclosure risk for socio-economic data, we first have to decide if the final data set should be publicly available (public-use file) or only available to a restricted group of people (scientific-use file). Public-use files should have lower disclosure risk than scientific-use files.

In any case, k-anonymity should be achieved. Additionally, we can look, for example, at the individual and household risks by considering the sampling weights (Sects. 3.5.1 and 3.6). In general, the sampling weights are often high for any household/person and not as wide-spread as in comparison to surveys from business statistics.

When considering, for example, the LFS data and the disclosure scenario with the 13 key variables above, we cannot provide k-anonymity for the combination of 13 key variables since it would result in too many suppressions. We may use a k-anonymity approach (e.g. $k = 5$) for 7 variables: degree of urbanization, sex, age, country of birth, years of residence, highest level of education and ILO working status. PRAM can be applied to the other identifying variables. This should ensure enough protection for at least producing scientific-use files.

7.5 The disclosure risk for business data should be discussed differently than for socio-economic data. The situation for business survey data is generally different, because large enterprises are often selected with probability one. Continuous key variables, such as turnover or number of employees, have to be heavily perturbed for large enterprises since the risk of disclosure is generally high for those enterprises. Remote access solutions may thus be the preferred solution for such data sets because the anonymized data will most likely have low data utility when using traditional SDC methods.

7.6 Let us discuss this with our previous selection of key variables: region, household size, gender, age, citizenship and educational level as categorical key variables, and the incomes as continuous key variables. In order to ensure k-anonymity, do the following:

- globally recode age into age classes,

- combine the 2–3 highest educational levels in case of low frequency,
- recode citizenship into fewer categories
- and delete or modify very large households (e.g. modify a 13-person household into a 9-person household by deleting 4 children).

After such recodings, local suppression will then ensure k-anonymity. In addition, a few suppressions should be made on observations with the highest disclosure risk (evaluated with the individual risk approach). Microaggregation can be applied on incomes, but with care: It should be applied on defined strata (e.g. children - non-children) and the zero patterns should be taken into account.

An alternative is the production of a synthetic SILC data, as done in Sect. 8.7.

Index

© Springer International Publishing AG 2017

M. Templ, *Statistical Disclosure Control for Microdata*,
DOI 10.1007/978-3-319-50272-4