
sdcMicro GUI manual Documentation

Thijs Benschop

Jan 04, 2019

TABLE OF CONTENTS

1	Introduction	3
1.1	What is sdcApp?	3
1.2	Statistical Disclosure Control (SDC)	3
1.3	What is the purpose of the manual?	3
1.4	Background literature on SDC for microdata	3
1.5	Outline of this guide	3
2	Installation and updating	5
2.1	Installing R and RStudio	5
2.2	sdcMicro package	7
2.3	Launching <i>sdcApp</i>	7
2.4	Updating R, RStudio and the sdcMicro package	8
2.5	Bug reporting on GitHub	10
3	Introduction to sdcApp	11
3.1	Starting sdcApp	11
3.2	Tab and panel structure	13
3.3	In-app help	14
3.4	Getting started	14
3.5	Set storage path	14
3.6	Quitting sdcApp	15
4	Loading and Preparing Data	17
4.1	Loading data	17
4.2	Inspect and explore data	19
4.3	Preparing data	20
5	Setup anonymization problem	23
5.1	Variable selection	23
5.2	Settings	25
5.3	Summary view	27
6	Risk measurement	29
6.1	Summary view	29
6.2	Detailed view	30
7	Anonymization methods	33
7.1	Recoding	33
7.2	k-Anonymity / local suppression	34
7.3	PRAM	36
7.4	Suppress values with high risk	38

7.5	Top/Bottom coding	38
7.6	Microaggregation	38
7.7	Adding noise	38
7.8	Rank swapping	38
7.9	Undo	38
8	Utility measurement	41
8.1	General utility measures in <i>sdApp</i>	41
8.2	Customized utility measures	41
9	Export data and reports	43
9.1	Export anonymized dataset	43
9.2	Exporting reports	43
10	Reproducibility	45
10.1	Exporting <i>R</i> script	45
10.2	Exporting reports	46
11	Undo	49
11.1	Undo one step	49
11.2	Undo several steps	49

This is documentation and guidance for *sdcApp*, a user interface for the *sdcMicro R* package, which provides tools for Statistical Disclosure Control (SDC) for microdata, also known as microdata anonymization.

INTRODUCTION

1.1 What is *sdcApp*?

sdcApp is the Graphical User Interface (GUI) for the R package *sdcMicro* (see [here](#)). The *sdcApp* opens the functionality of *sdcMicro* to users not familiar with the statistical programming language R. *sdcMicro* is an add-on package for the statistical software R for Statistical Disclosure Control (SDC) of microdata and includes functions for risk measurement, anonymization and utility measurement for microdata. All functionality available in the *sdcMicro* package is also available in *sdcApp*.

1.2 Statistical Disclosure Control (SDC)

A large part of the data collected by statistical agencies cannot be published directly due to privacy and confidentiality concerns. These concerns are both of legal and ethical nature. SDC seeks to treat and alter the data so that the data can be published or released without revealing the confidential information it contains, while, at the same time, limit information loss due to the anonymization of the data. There are two strands of literature on SDC: 1) for microdata and 2) for tabular data. *sdcMicro* and *sdcApp* only provide the tools and methodology for protecting microdata.

1.3 What is the purpose of the manual?

This manual is designed to provide step-by-step guidance through the process of anonymizing a dataset with microdata. Both the background information on methods and measures is provided as well as instructions on how to complete these steps in *sdcApp*. As *sdcApp* is a GUI for the *sdcMicro* package, users familiar with using R for statistical analysis may prefer to carry out the anonymization process using R from command-line. More information and guidance on using *sdcMicro* from command-line is available in the SDC Practice Guide available [here](#). This guide also provides more detailed background information on SDC.

1.4 Background literature on SDC for microdata

Add here links to websites/books

1.5 Outline of this guide

This guide is divided into the following main sections:

1. the Section [Installation and updating](#) guides the user through the installation process of sdcApp, which includes the installation of R, RStudio as well as the required packages. It also discusses the need and process of regular updates of all software components.
 2. the Section [Introduction to sdcApp](#) covers how to launch and close the application and provides a brief overview of structure of the application. of the structure of the application
 3. the Section [Loading and preparing data](#) describes how to load microdata into the application. It also discusses the requirements to the
 4. the Section [Setup anonymization problem](#) covers the variable selection and setup of an SDC problem.
 5. the Section [Risk measurement](#) covers methods to measure the disclosure risk in the microdata.
 6. the Section [Anonymization methods](#) covers anonymization methods for quantitative and qualitative variables.
 7. the Section [Utility measurement](#) covers the measurement of information loss resulting from anonymization of the data
 8. the Section [Export data and reports](#) describes how to export the anonymized dataset and generate reports.
 9. the Section [Reproducibility](#) covers functionality that render the anonymization process reproducible.
- ? Add case study ?

INSTALLATION AND UPDATING

This section will guide you through the steps you need to take to install *sdcApp*. *sdcApp* is a graphical user interface for the *sdcMicro* package. The *sdcMicro* package is an add-on package for the statistical software R. In order to start working with *sdcApp*, you need to install R, RStudio¹ as well as several add-on packages for R. All software is available free of charge and open-source. R and RStudio run on most platforms, including Windows, Mac OS X and Linux. To use *sdcApp*, a webbrowser needs to be installed as well.

R, RStudio, the *sdcMicro* package as well as dependencies are regularly updated. Therefore, it is recommended to regularly update to the latest version of the software. The Section *Updating R, RStudio and the sdcMicro package* shows how to check for updates and install updates.

2.1 Installing R and RStudio

The first step in the installation of *sdcApp* is the installation of R and RStudio. The free open source statistical software R can be downloaded from the [CRAN website](#). By selecting your OS, the installer will be downloaded to your computer. In order to install R, open the installer and follow the installation steps

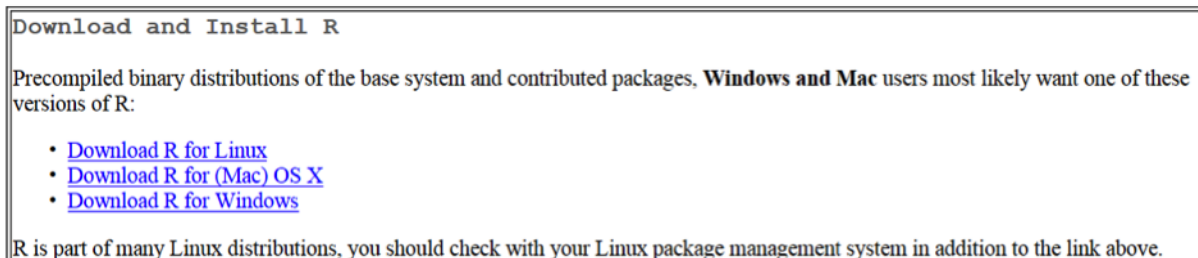


Fig. 2.1: Select OS to start downloading R

Once R is successfully installed, we can install RStudio. RStudio is an IDE (integrated development environment) for R. RStudio makes working with R much easier. RStudio Desktop and RStudio Server can be downloaded free of charge from the [RStudio website](#). On this webpage, scroll down to the overview of different versions as shown in Figure and select the version corresponding to your OS to start downloading. In order to install RStudio, open the installer and follow the installation steps.

Note: We recommend updating to the latest versions of R and RStudio if this software is already installed on your computer before moving on. See also the Sections *Updating R* and *Updating RStudio* for more information on updating the software.

¹ Technically speaking, RStudio is not required to run *sdcApp*. Nevertheless, we recommend to install RStudio for a better user experience.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.1.456 - Windows Vista/7/8/10	85.8 MB	2018-07-19	24ca3fe0dad8187aabd4bfb9dc2b5ad
RStudio 1.1.456 - Mac OS X 10.6+ (64-bit)	74.5 MB	2018-07-19	4fc4f4f70845b142bf96dc1a5b1dc556
RStudio 1.1.456 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB	2018-07-19	3493f9d5839e3a3d697f40b7bb1ce961
RStudio 1.1.456 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB	2018-07-19	863ae806120358fa0146e4d14cd75be4
RStudio 1.1.456 - Ubuntu 16.04+/Debian 9+ (64-bit)	64.9 MB	2018-07-19	d96e63548c2add890bac633bdb883f32
RStudio 1.1.456 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB	2018-07-19	1df56c7cd80e2634f8a9fdd11ca1fb2d
RStudio 1.1.456 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB	2018-07-19	5e77094a88fdbddddd0d35708752462

Fig. 2.2: Select version to start downloading RStudio

Once R and RStudio are installed on your computer, open RStudio. The RStudio interface consists of four different panes as shown in Figure.

1. script editor (by default left up)
2. (by default right up)
3. R console (by default left down)
4. (by default right down)

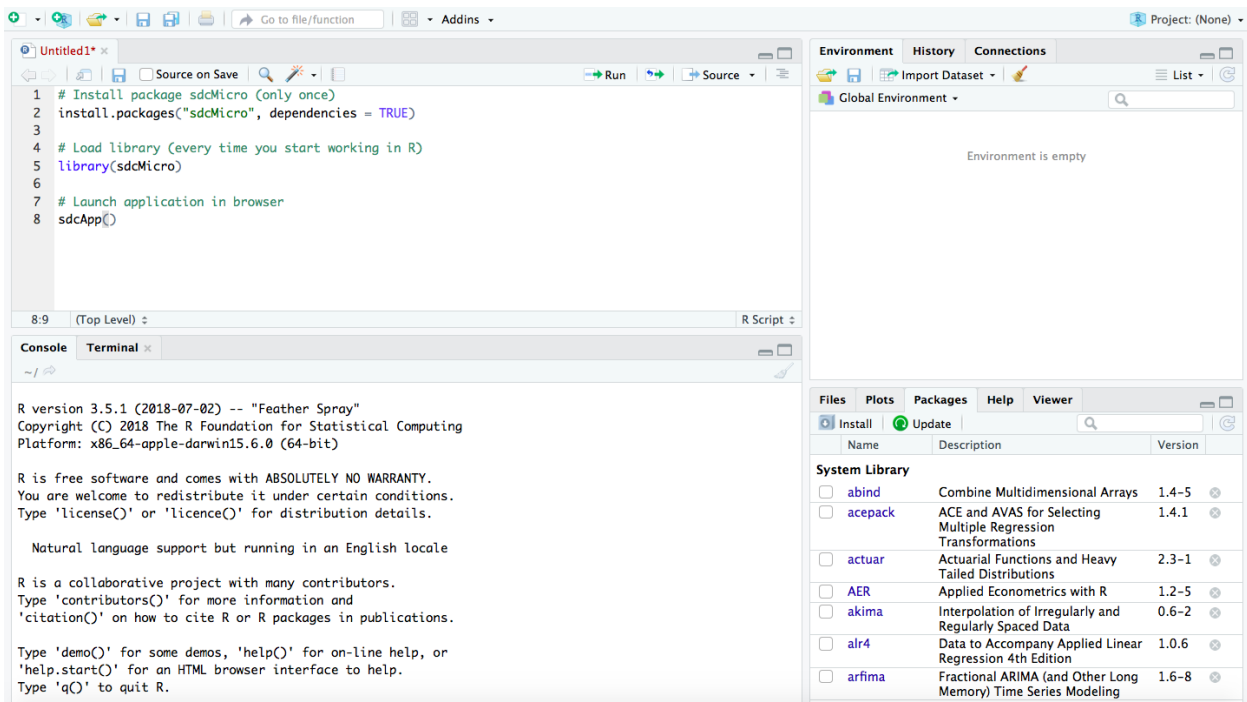


Fig. 2.3: Screenshot RStudio

2.2 sdcMicro package

sdcApp is included in the R package *sdcMicro*. Once RStudio is open, *sdcMicro* can be installed by executing commands in the R console. *sdcMicro* and a set of other R packages that are required by the *sdcMicro* package are downloaded from the CRAN servers. Therefore, it is necessary to be connected to the internet during the installation process.²

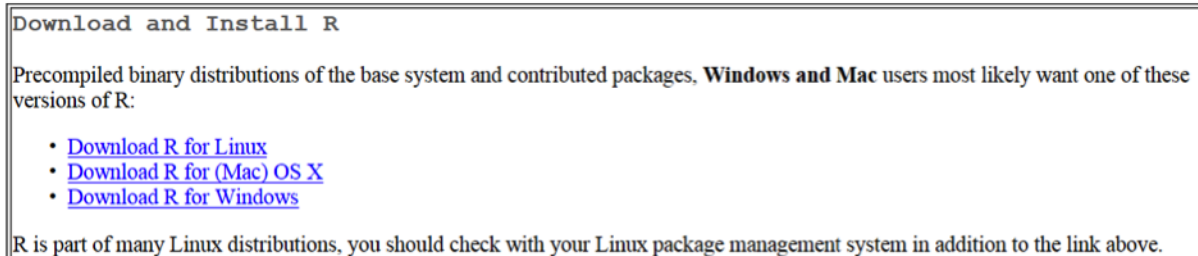


Fig. 2.4: Screenshot RStudio

In order to install the latest version of the *sdcMicro* package, type the command `install.packages("sdcMicro", dependencies = TRUE)` in the console and press enter to execute. The first time you are installing R packages, a prompt will ask you to select a CRAN mirror (server) to install the package from. Since the packages on all mirrors are identical, you can choose any of the locations. The *sdcMicro* package itself uses functionality from a set of other R packages (e.g., *haven* for reading files in different formats). By specifying the `dependencies` argument to `TRUE`, these dependencies will automatically be installed too.

Listing 2.1: Installing sdcMicro package

```
1 # install sdcMicro package
2 install.packages("sdcMicro", dependencies = TRUE)
```

Note: Also dependencies will be installed and the installation may take some time. Dependencies are other add-on packages, of which the functionality is required to run the *sdcMicro* package.

Note: An internet connection is not required while using *sdcMicro* and *sdcApp* and the data are stored locally on your computer or server. The web browser uses a local host IP, which is not connected to the internet and the browser is only used to communicate with the running R session.

2.3 Launching sdcApp

Once the *sdcMicro* package is successfully installed, the *sdcMicro* package needs to be loaded. Installing the package is only required once (except for updating), whereas loading the package is required every time a new R session is started.

You can load the *sdcMicro* package by typing `library(sdcMicro)` and launch the application by typing `sdcApp()`.

sdcApp opens in your system's default web browser through the local host IP `127.0.0.1`. *sdcApp* works with recent versions of any web browser. Due to small issues encountered with some browsers, we recommend to use

² It is possible to download R, RStudio and the packages and transfer the files to the computer with for example a USB drive in case the computer *sdcMicro* should be installed on cannot be connected to the internet for technical or confidentiality reasons.

Google Chrome, Mozilla Firefox or Safari for the best performance. In case your default web browser is not one of the aforementioned browsers, you can simply open an alternative browser and copy paste the internal IP address to the new browser. *sdcApp* will open in the new browser.

Furthermore, it's important that your R session is allowed to use the installed webbrowser.



```

> library(sdcMicro)
-----
This is sdcMicro v5.3.0.
For references, please have a look at citation('sdcMicro')
Note: since version 5.0.0, the graphical user-interface is a shiny-app that can be started with sdcApp().
Please submit suggestions and bugs at: https://github.com/sdcTools/sdcMicro/issues
-----
> sdcApp()

Attaching package: 'DT'

The following objects are masked from 'package:shiny':

    dataTableOutput, renderDataTable

data.table 1.11.8 Latest news: r-datatable.com

Listening on http://127.0.0.1:7970
|

```

Fig. 2.5: R console with local IP after launching *sdcApp*

Listing 2.2: Loading *sdcMicro* package and launching *sdcApp*

```

1 # Load sdcMicro package
2 library(sdcMicro)
3
4 # Launch sdcApp (opens in browser window)
5 sdcApp()

```

In rare cases, not all dependencies are correctly installed and the following error message appears in the R console upon loading the *sdcMicro* package.

Please install the package(s) indicated in the error message manually by using the command `install.packages()` with the name of the package(s). In the example error message, this would be for the packages .

2.4 Updating R, RStudio and the *sdcMicro* package

R, RStudio, the *sdcMicro* package as well as dependencies are regularly updated. Updates include bug fixes as well as additional functionality. Therefore, it is recommended to regularly update to the latest version of the software.

2.4.1 Updating R

RStudio uses by default the most recent version of R available on your system. New versions of R packages, including the *sdcMicro* package, rely on the newest version of R. Therefore, it's important to regularly check for updates of R. The easiest way to do so is to visit regularly the [CRAN website](https://cran.r-project.org/). If a new version of R is available, the same steps as as

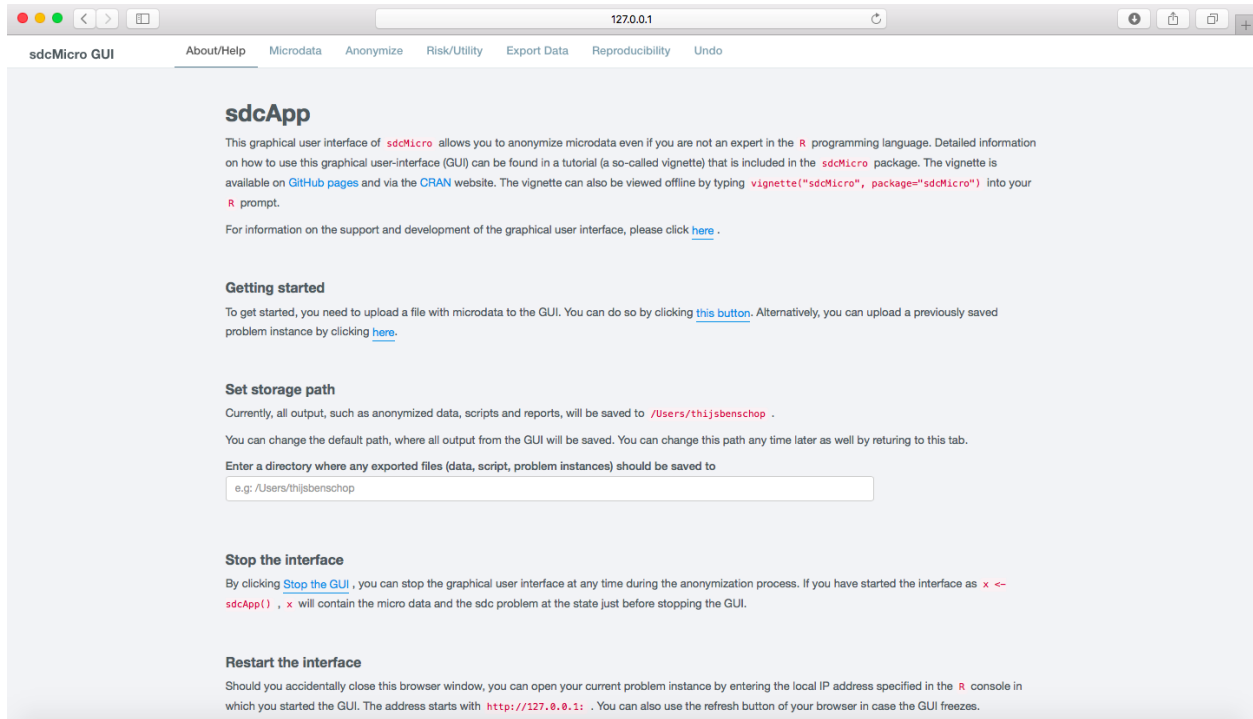


Fig. 2.6: Start screen sdcApp in browser with local IP

for the installation of R need to be followed as described in the Section *Installing R and RStudio*. The version number of the R version installed on your computer appears in the R console upon launching R or RStudio (cf. Figure).

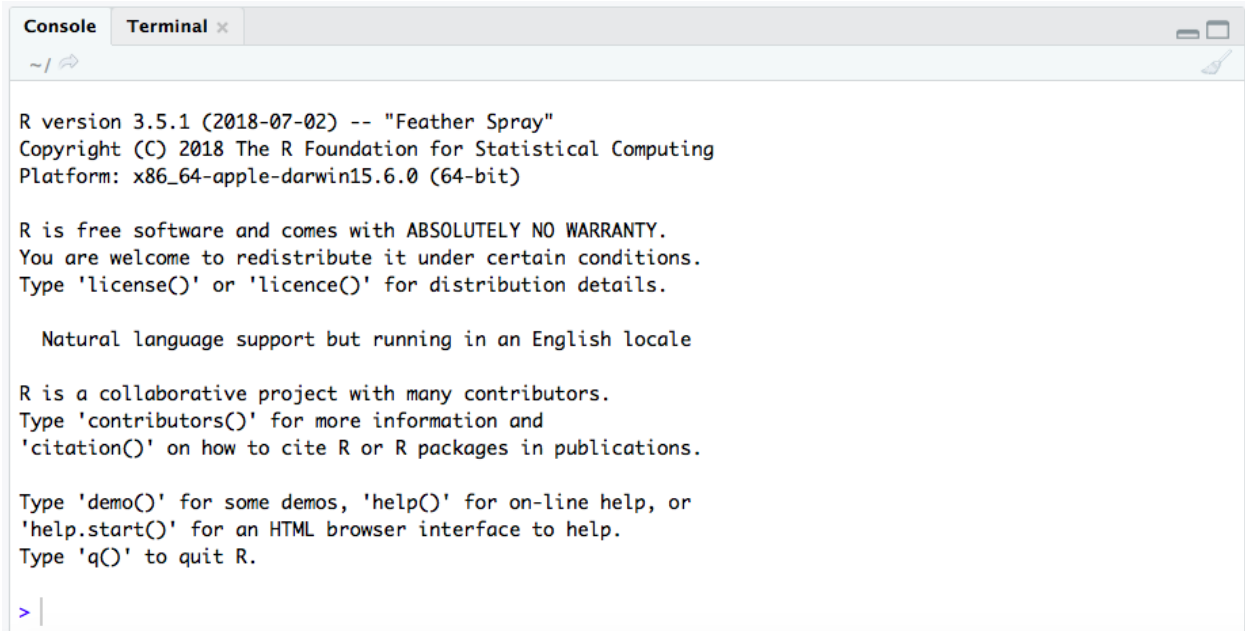
2.4.2 Updating RStudio

RStudio has Help -> Check for updates. If an update is available, the current version number and the newest version number are shown. In order to install the newer version, you need to visit the [RStudio](#) website and follow the steps as described in the Section *Installing R and RStudio*.

2.4.3 Updating R packages

The sdcMicro package is regularly updated to fix bugs and add functionality. In order to check for newer versions, click on the Update button to get an overview of all packages that have newer versions available. By clicking Select all, these packages are all automatically updated.

Report a bug



```
Console Terminal x
~/
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Fig. 2.7: R console with version number

Listing 2.3: Updating packages

```
1 # Update
2 install.packages()
```

2.5 Bug reporting on GitHub

The sdcMicro package is open source software and the source code can be easily viewed on the [GitHub](#) of the sdcMicro project. There you can also report alleged bugs and raise other issues.

INTRODUCTION TO SDCAPP

sdcApp is a user-friendly application for microdata anonymization and is built on the [Shiny](#) technology. Shiny allows users to communicate through a GUI that runs in a webbrowser with a local *R* session. The local *R* session performs the necessary calculations. In the case of *sdcApp*, most functionality used in *R* is included in the [sdcMicro](#) package.

sdcApp has a tab structure and consists of seven tabs, which in turn consist of up to three panels. This structured is further explored below. The tabs and panels are used to navigate through the app.

3.1 Starting *sdcApp*

After succesful installation (see the Section [Installation and updating](#)), *sdcApp* is ready for use. Every single time *sdcApp* is used, first the applications *R* or *RStudio* need to be opened. We recommend to use *RStudio* for ease of use. After launching *R* or *RStudio*, the *sdcMicro* package needs to be loaded and *sdcApp* needs to be launched. To load *sdcMicro* and launch *sdcApp*, enter the code as shown in [Listing 3.1](#) in the *R* console. Press enter after each line to execute the line of code. [Fig. 3.1](#) shows the output in the *R* console after successfully launching *sdcApp*.

Listing 3.1: Loading *sdcMicro* package and launching *sdcApp*

```
1 # Load sdcMicro package
2 library(sdcMicro)
3
4 # Launch sdcApp (opens in browser window)
5 sdcApp()
```

Note: You can omit the lines starting with a hash tag (#) as these are comment lines and ignored by the *R* interpreter.

The application opens in a new tab in your default web browser. In case you prefer to use an alternative web browser, you can simply copy the address of the localhost and paste it into a different browser on the same machine. The localhost address can be found in the output in the *R* console. The address starts with `http://127.0.0.1:` followed by a four digit number. In the example in [Fig. 3.1](#), the full localhost address is `http://127.0.0.1:3256`. The application opens on the **About/Help** tab (see [Fig. 3.2](#)).

Note: Firewalls and other settings on your computer and browser may prevent *sdcApp* from opening in your web-browser. As a first thing you could try to copy paste the localhost address into your webbrowser. If that is not successful, try changing the settings of your browser and firewall.

```

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(sdcMicro)
-----
This is sdcMicro v5.3.0.
For references, please have a look at citation('sdcMicro')
Note: since version 5.0.0, the graphical user-interface is a shiny-app that can be started with sdcApp().
Please submit suggestions and bugs at: https://github.com/sdcTools/sdcMicro/issues
-----
> sdcApp()

Attaching package: 'DT'

The following objects are masked from 'package:shiny':

    dataTableOutput, renderDataTable

data.table 1.11.8 Latest news: r-datatable.com

Listening on http://127.0.0.1:3256
  
```

Fig. 3.1: R console after loading the *sdcMicro* package and launching *sdcApp*

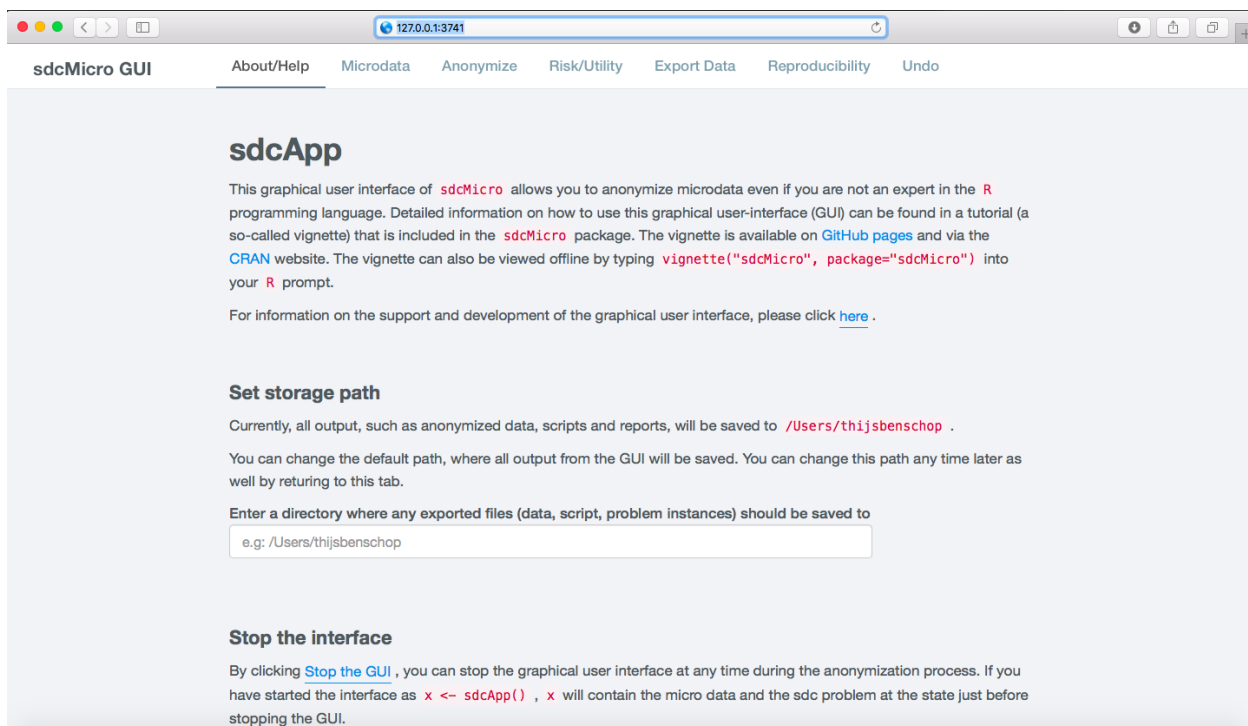


Fig. 3.2: **About/Help** tab in web browser after launching *sdcApp* with localhost address

3.2 Tab and panel structure

The sdcApp consists of seven different tabs that serve different parts of the SDC process. The tabs can be selected in the navigation bar at the top of the page (cf. the area indicated with 1 in Fig. 3.3). The navigation bar is visible at all times. The content of each tab may change as function the specified SDC problem and the current state of the SDC process. For example, anonymization methods for continuous key variables are not shown on the **Anonymize** tab, if no continuous key variables are selected.

- **About/Help** Landing page to set storage path, quit and restart *sdcApp* as well as provide feedback to the developers
- **Microdata** Page to load, view, explore and prepare the microdata to be anonymized
- **Anonymize** Page to setup the anonymization problem (select variables, set parameters). Once the problem is defined, this page shows a summary of the anonymization problem and allows to apply anonymization methods
- **Risk/Utility** Page to evaluate disclosure risk and information loss (data utility)
- **Export Data** Page to export the anonymized data and reports on the anonymization process
- **Reproducibility** Page with functionality to guarantee the reproducibility of the process by exporting the *R* script or problem instance
- **Undo** Page to revert one or several steps in the anonymization process

Each tab consists of two panels: the left sidebar (cf. the area indicated with 2 in Fig. 3.3) and the main panel (cf. the area indicated with 3 in Fig. 3.3). The left panel allows the user to navigate between different function on the same tab, e.g., different risk measures. Some tabs have an additional right sidebar (cf. the area indicated with 4 in Fig. 3.3), which provide summary information on the current SDC problem.

The screenshot displays the sdcMicro GUI with the Risk/Utility tab selected. The top navigation bar (1) contains tabs: About/Help, Microdata, Anonymize, Risk/Utility, Export Data, Reproducibility, and Undo. The left sidebar (2) lists various functions under three categories: Risk measures, Visualizations, and Numerical risk measures. The main panel (3) shows a slider for selecting a value for 'k' (set to 3) and a table of 10 observations violating 3-anonymity. The right sidebar (4) includes a 'Variable selection' table and an 'Additional parameters' table.

Variable name	Type	Additional suppressions by local suppression algorithm
urbrur	cat. key variable	0
sex	cat. key variable	0
age	cat. key variable	0
sampling_weight	sampling weight	

Parameter	Value
number of records	4580
alpha	1

Fig. 3.3: Risk/Utility tab with navigation bar and panel structure

3.3 In-app help

By hovering over the *i* icon in *sdcApp*, additional information on e.g., specific parameters and the interpretation of results is provided. The help information is mainly intended to provide a brief reminder and is not meant to replace a thorough study of the SDC literature on risk and utility measurement and anonymization methods. Fig. 3.4 shows the help pop-up for the variable selection table.

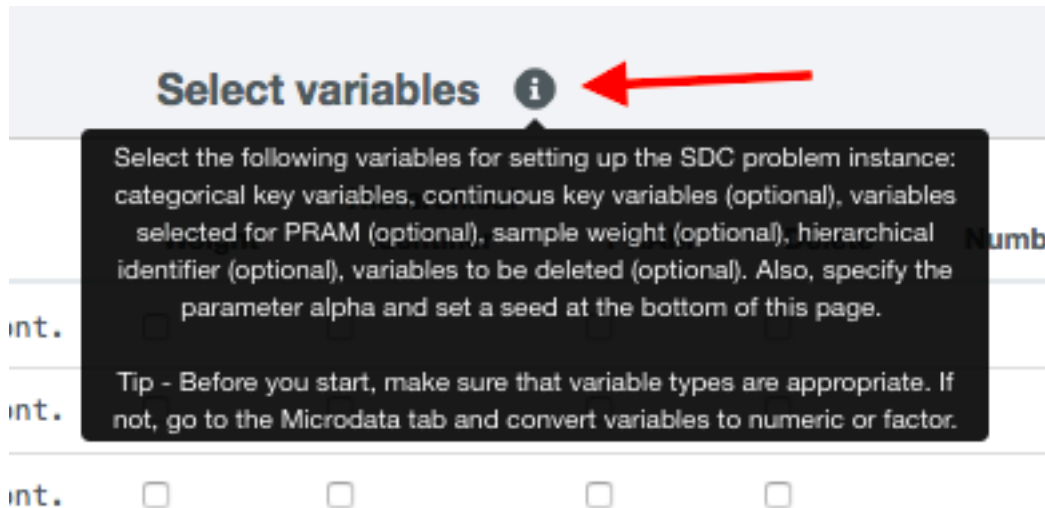


Fig. 3.4: Help pop-up when moving with mouse cursor over *i* icon

3.4 Getting started

Use testdata dataset: all examples in this guide are illustrated with the testdata dataset.

3.5 Set storage path

All output exported from *sdcApp*, such as the anonymized dataset, reports and scripts will be saved in the directory shown under the header **Set storage path** on the **About/Help** tab (cf. Fig. 3.5). Upon launching *sdcApp*, this directory is set to the *R* working directory. Change the working directory to a the folder in the project directory with the dataset to be anonymized by typing the path to this folder in the input box (cf. Fig. 3.5). Once a valid path on your computer is entered, click the blue button **Update the current output path** to change the path. If the entered path is not a valid path on your system, a red button appears with the text **The specified directory does not exist, thus the path can't be updated**. It is recommended to create a new folder in the project directory for the *sdcApp* output. The file names of the output files contain a date and time stamp as well as a brief description, e.g., exported-Data_sdcMicro_20181010_1211.dta for the anonymized microdata in STATA format on October 10, 2018 at 12:11 and exportedProblem_sdcMicro_20180304_1633.rdata for the saved problem instance as *R* datafile on March 4, 2018 at 16:33.

Note: The storage path to the output folder needs to be specified every time *sdcApp* is launched.

Note: If an *sdcProblem* is saved and reloaded, the storage path is set to the path saved in the *sdcProblem*. If the

problem is loaded on a different computer than it was saved at, the storage path may be invalid and needs to be updated in the same way as described above.

Set storage path

Currently, all output, such as anonymized data, scripts and reports, will be saved to `/Users/thijsbenschoop`.

You can change the default path, where all output from the GUI will be saved. You can change this path any time later as well by returning to this tab.

Enter a directory where any exported files (data, script, problem instances) should be saved to

Update the current output path

Fig. 3.5: View and set storage path for file export

3.6 Quitting sdcApp

To quit *sdcApp*, click on **Stop the GUI** under the header **Stop the interface** on the **About/Help** tab. It is recommended to quit *R* or *RStudio* after quitting *sdcApp* to ensure that nothing is left in the memory. This especially applies to a restart due to *sdcApp* not responding.

Stop the interface

By clicking [Stop the GUI](#), you can stop the graphical user interface at any time during the anonymization process. If you have started the interface as `x <- sdcApp()`, `x` will contain the micro data and the sdc problem at the state just before stopping the GUI.

Fig. 3.6: Button to quit *sdcApp* on the **About/Help** tab

Save SDC problem to continue working later. Possible once the SDC problem is defined. See undo section

LOADING AND PREPARING DATA

This section discusses how to load microdata into *sdcApp* and prepare the data for the SDC process.

The first step in the SDC process is loading the dataset into *sdcApp*. *sdcApp* supports most common data formats, such as *R*, *STATA*, *SPSS* and *SAS* files. First time users may also load one of the two practice datasets, which are included in *sdcApp*, to explore *sdcApp* and methods. Most examples in this guide are illustrated by using the practice dataset *testdata* and can be reproduced.

After loading the data, the user needs to prepare the data for the SDC process. Most preparation steps can be carried out in *sdcApp*, although users may find it more convenient to perform some of these actions in another statistical software before loading the data in *sdcApp*.

4.1 Loading data

4.1.1 Testdata

sdcApp includes two practice datasets: *testdata* and *testdata2*. The dataset *testdata* is used to illustrate methods and examples in this guide. In order to replicate these examples, the user needs to load this dataset. In order to load the *testdata* dataset, navigate to the **Microdata** tab and select **Testdata/Internal data** in the left sidebar. Select the dataset from the dropdown menu and click the button **Load data**. This is illustrated in Fig. 4.1. After loading the *testdata* dataset, the loaded dataset is displayed (cf. Fig. 4.2).

Any other datasets loaded in the current *R* session are also shown in the list with available datasets and can be loaded.

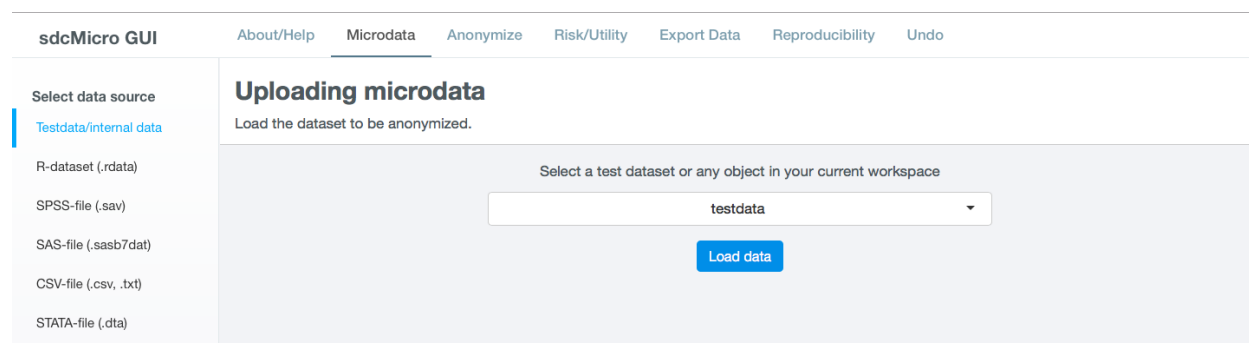


Fig. 4.1: Load testdata on **Microdata** tab

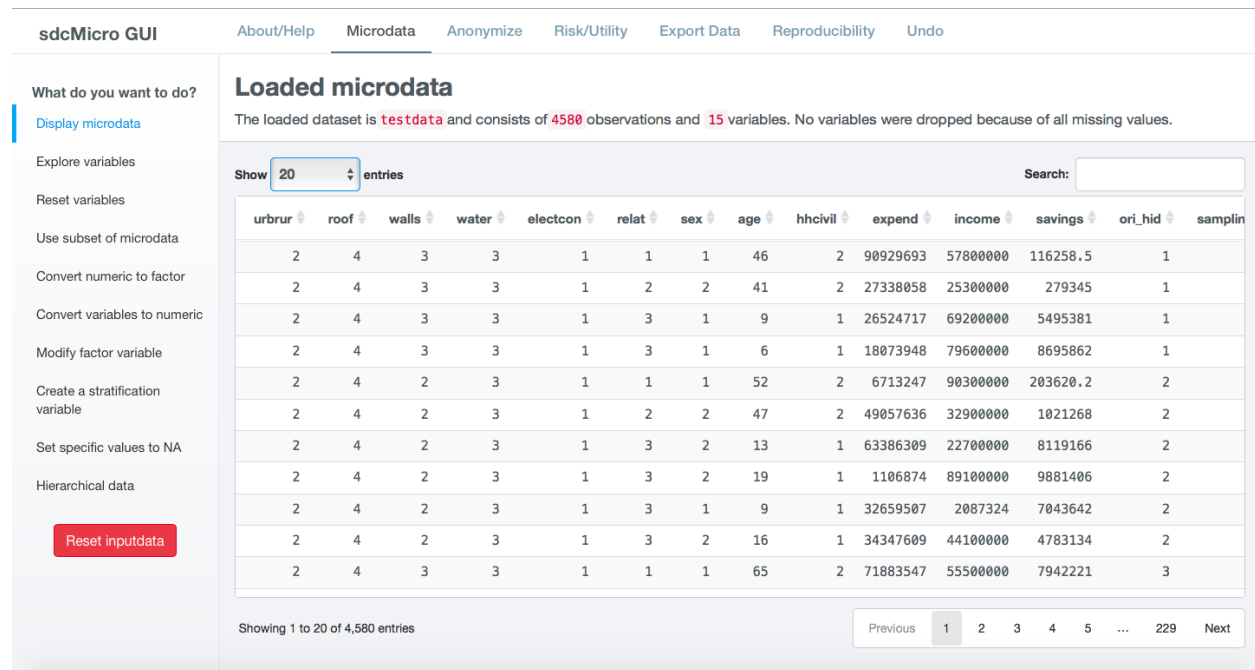


Fig. 4.2: Loaded dataset

4.1.2 Other microdata

sdcApp supports datasets in several foreign data formats (cf. Table 4.1). If the microdata is not in one of these data formats, another software can be used to convert the data, such as *Stat/Transfer*. Also some statistical software allow to export the data in another data format.

Table 4.1: Data formats compatible with *sdcApp*

Software	File extension
R/RStudio	.rdata
SPSS	.sav
SAS	.sas7bdat
CSV	.csv, .txt
STATA	.dta

In order to load a dataset, select the corresponding data format from the left sidebar of the Microdata tab. For all formats the user can set two options:

- Convert string variables (character vectors) to factor variables?** If **TRUE** (default), variables of type string are automatically converted to categorical variables (type factor in *R*). Categorical variables need to be of type factor in *sdcApp*. Remove any textual variables, such as ‘Specify other:’ variables before loading the data. These variables are oftentimes not suitable for release and require long computation times to be transformed to factor. If **FALSE**
- Drop variables with only missing values (NA)?** If **TRUE** (default), variables that contain only missing values (NA in *R*) are removed upon loading the data. This does not cause any loss of information, as these variables do not contain information. However, variables with only missing values can cause issues in *sdcApp*. If **FALSE**, no variables are deleted.

If the selected data format is a CSV-file, two additional options need to be specified:

1. **Does the first row contain the variable names?** If `TRUE`, the values in the first row are used as variable names. If `FALSE`, the variables names are set to `V1`, `V2`, `V3`, ... in the order of appearance in the dataset.
2. **Field separator** The field separator in the csv file needs to be specified. Options are comma (,), semicolon (;) and tab.

After setting the options for the data upload, click on the button **Browse** to access the file system in your computer and select the microdata file. The file is upload immediately after selection. After loading the file, which may

Note: Set the additional options before selecting the datafile from your file system. Upon selection after clicking **Browse**, the file is immediately loaded and settings can no longer be changed. If the file was accidentally loaded before setting all parameters, the file needs to be reloaded after first resting the microdata by clicking **Reset microdata** in the left sidebar.

Note: The default maximum file size in *sdcApp* is 50 MB. In order to upload larger files, the maximum file size in MB needs to be specified upon launching *sdcApp*. This can be achieved by specifying the argument `maxRequestSize`:

Listing 4.1: Launching *sdcApp* to load larger files

```
1 # Launch sdcApp with increased max. file size (200MB)
2 sdcApp(maxRequestSize = 200)
```

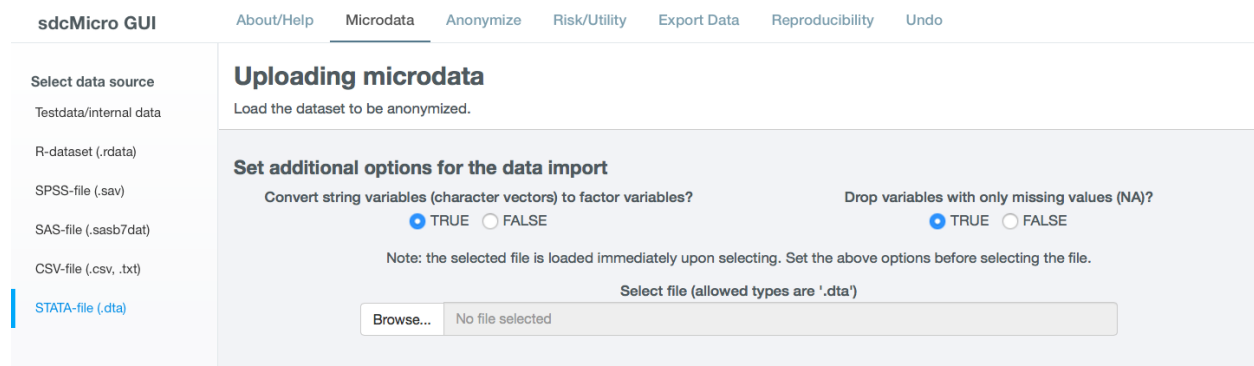


Fig. 4.3: Load data on Microdata tab - example STATA dataset

After loading the dataset, the data is shown in the **Microdata** tab. The **Microdata** tab changes and the functionality for loading microdata is replaced with functionality to explore and prepare the dataset (cf. Fig. 4.4). The left sidebar shows different options to explore and prepare the data for the anonymization process, as discussed in the next sections.

4.2 Inspect and explore data

After loading the dataset into *sdcApp*, the data is shown on the Microdata tab. At the top of the data viewer, the number of observations and variables is shown as well as the number of variables that were deleted as a result all missing values (cf. Fig. 4.4).

Note: If *Drop variables with only missing values (NA)?* is set to `TRUE`, the number of variables shown may be lower

sdcMicro GUI

About/Help Microdata Anonymize Risk/Utility Export Data Reproducibility Undo

What do you want to do?

Display microdata

Explore variables

Reset variables

Use subset of microdata

Convert numeric to factor

Convert variables to numeric

Modify factor variable

Create a stratification variable

Set specific values to NA

Hierarchical data

Reset inputdata

Loaded microdata

The loaded dataset is **testdata** and consists of **4588** observations and **15** variables. No variables were dropped because of all missing values.

Show 20 entries

Search:

urbrur	roof	walls	water	electcon	relat	sex	age	hhcivil	expend	income	savings	ori_hid	sampling_weight	household_weights
2	4	3	3	1	1	1	46	2	90929693	57800000	116258.5	1	100	25
2	4	3	3	1	2	2	41	2	27338858	25300000	279345	1	100	25
2	4	3	3	1	3	1	9	1	26524717	69200000	5495381	1	100	25
2	4	3	3	1	3	1	6	1	18873948	79600000	8695862	1	100	25
2	4	2	3	1	1	1	52	2	6713247	90300000	283620.2	2	100	16.6666666666667
2	4	2	3	1	2	2	47	2	49857636	32900000	1021268	2	100	16.6666666666667
2	4	2	3	1	3	2	13	1	63386309	22700000	8119166	2	100	16.6666666666667
2	4	2	3	1	3	2	19	1	1106874	89100000	9881406	2	100	16.6666666666667
2	4	2	3	1	3	1	9	1	32659507	2087324	7043642	2	100	16.6666666666667
2	4	2	3	1	3	2	16	1	34347609	44100000	4783134	2	100	16.6666666666667
2	4	3	3	1	1	1	65	2	71883547	55500000	7942221	3	100	33.3333333333333
2	4	3	3	1	2	2	60	2	55174345	41200000	4318171	3	100	33.3333333333333
2	4	3	3	1	5	2	6	1	46002021	99600000	2680967	3	100	33.3333333333333
2	4	3	3	1	1	1	34	2	33042094	98400000	3662611	4	100	33.3333333333333
2	4	3	3	1	2	2	31	2	22328588	68900000	6668614	4	100	33.3333333333333

Showing 1 to 20 of 4,588 entries

Previous 1 2 3 4 5 ... 229 Next

Fig. 4.4: Microdata tab after loading dataset

than the number of variables in the loaded dataset.

It is important to check whether the data was imported completely and correctly by browsing the dataset in *sdcApp*. If, for example, records are missing or labels are corrupted, then these issues need to be fixed outside of *sdcApp* and the data need to be reimported.

By clicking **Explore variables** in the left sidebar, univariate and bivariate summary statistics appropriate for the variable type can be displayed. If one variable is selected, univariate summary statistics are shown.

Note: The choice of summary statistics is based on the variable type specified in *R* (shown in brackets after the variable name, e.g., *urbrur* (integer)). Therefore, the representation may not be correct, if the variable type does not correspond with the variable content. By converting the variable (see *Convert variable type*), the correct summary statistics will be displayed.

4.3 Preparing data

Most datasets need to be prepared before the start of the anonymization process. Examples of data preparation are removing variables that are not suitable for release, etc. It is recommended to carry out the data preparation in a statistical software of choice, before loading the data in *sdcApp*. Data preparation includes

After loading the data in *sdcApp*, still some steps may need to be carried, which are specific to the needs of the *sdcApp*. These steps are discussed in the following subsections.

4.3.1 Convert variable type

numeric to factor

to numeric

4.3.2 Set specific values to NA

Missing values play an important role in anonymization of microdata. In particular when measuring disclosure risk of categorical key variables (see ‘Risk’ [___](#)). sdcApp only considers the R missing value NA as missing. Therefore, it is important to recode other missing values, such as 9, 99, 998 or 999, “Missing”, “Not applicable” after loading the data to the R missing value NA, if appropriate. Many standard missing value codes in the data, such as . in STATA are automatically converted to NA upon loading the data into *sdcApp*.

sdcMicro GUI About/Help Microdata Anonymize Risk/Utility Export Data Reproducibility Undo

What do you want to do?

- Display microdata
- Explore variables
- Reset variables
- Use subset of microdata
- Convert numeric to factor
- Convert variables to numeric
- Modify factor variable
- Create a stratification variable
- Set specific values to NA**
- Hierarchical data

Set missing values to NA

The loaded dataset may contain different missing value codes, such as 9, 999, -9, etc. sdcMicro can only interpret missing values that are coded NA. Here you can set other missing value codes to NA (by missing value code). All records with that value in the variable will be recoded to NA. Alternatively one can set individual cells to NA by selecting the variable and record id (by record ID). Note that it is impossible to later retrieve the original missing values. R does not allow for distinct missing value codes.

How to select values to recode to NA?
☒ by value ☐ by record ID

Select variable: walls (integer) Select value to recode to NA: 9

Set values to NA

Show 10 entries

	id	urbrur	roof	walls	water	electcon	relat	sex	age	hhcivil	expend	income	savings	ori_hid
1	1	2	4	3	3	1	1	1	46	2	90929693	57800000	116258.5	
2	2	2	4	3	3	1	2	2	41	2	27338058	25300000	279345	

Reset input data

Fig. 4.5: Screen to set specific value in a variable to NA

4.3.3 Modify factor variable

Recoding (see Recoding)

4.3.4 Create stratification variable

sdcMicro GUI About/Help Microdata Anonymize Risk/Utility Export Data Reproducibility Undo

What do you want to do?

- Display microdata
- Explore variables
- Reset variables
- Use subset of microdata
- Convert numeric to factor
- Convert variables to numeric
- Modify factor variable
- Create a stratification variable**
- Set specific values to NA
- Hierarchical data

Create a stratification variable

Many SDC methods can be applied within strata. Here you can generate a new stratification variable by chaining together values of two or more variables. The number of strata is the product of the number of factor levels in the selected variables. For instance by choosing gender (male, female) and region (region 1, region 2), 4 strata are generated (male - region 1, male - region 2, female - region 1, female - region 2). By default the variable name of the stratification variable consists of the variable names separated by '_'. You can also specify the variable name by typing it into the text field. The new variable is added to the loaded micro data set and will be exported.

Select variables to generate a stratification variable: urbrur (integer) roof (integer) Specify variable name for stratification variable: urbrur_roof

Create stratification variable

Reset input data

Fig. 4.6: Screen to create new stratification variable

4.3.5 Reset variables

4.3.6 Hierarchical data

The screenshot shows the 'sdcMicro GUI' interface with the 'Microdata' tab selected. The left sidebar lists various actions, with 'Hierarchical data' highlighted. The main panel is titled 'Deal with hierarchical Data' and contains the following elements:

- What do you want to do?**: Two radio buttons. The first, 'Prepare file for the anonymization of household level variables', is selected. The second is 'Merge an anonymized household level file into the full dataset'.
- Prepare household-level data**: A section explaining that one record per household is selected and non-household variables are removed. It notes the importance of selecting household sampling weights.
- Select the household id variable**: A dropdown menu showing 'ori_hid (integer)'.
- Please select all variables that refer to households and not to individuals**: A text input field containing 'household_weights'.
- Buttons**: A red 'Reset inputdata' button and a blue 'Create household-input data' button.

Fig. 4.7: Screen to create household level dataset

The screenshot shows the same 'sdcMicro GUI' interface, but with the 'Merge Data' section active. The left sidebar remains the same. The main panel contains the following elements:

- What do you want to do?**: Two radio buttons. The first is 'Prepare file for the anonymization of household level variables'. The second, 'Merge an anonymized household level file into the full dataset', is selected.
- Merge Data**: A section explaining that an already exported and anonymized household-level file will replace the household level variables in the raw dataset.
- Note**: A text note stating 'the selected file is loaded immediately. Set options before selecting the file.'
- Select File (allowed types are '.rdata')**: A file selection interface with a 'Browse...' button and a text field showing 'No file selected'.

Fig. 4.8: Screen merge anonymized household level dataset with individual level dataset

4.3.7 Use subset of microdata

SETUP ANONYMIZATION PROBLEM

Based on the analysis of the disclosure scenarios (see), the user needs can make the variable selection in *sdApp* and set some other parameters in order to define the so-called SDC problem. Once the data is loaded and prepared, the tab *Microdata* shows a variable selection matrix in the main panel. The right sidebar shows several parameter settings and allows to have a quick summary view of each of the variables in the loaded dataset.

5.1 Variable selection

In order to setup an SDC problem the user needs to make a variable selection. The variable selection itself is the result of the analysis of disclosure scenarios and is beyond the scope of this manual. We refer to Chapter in for a thorough discussion of disclosure scenarios.

The matrix shown in Fig. 5.1 contain one row for each variable in the loaded dataset and nine different columns as described in Table 5.1. The user can select for each variable the function it has in the sDC problem. No selection needs to be made for variables that are not relevant to the anonymization process and can be released without further treatment. Each of the different columns is described in more detail:

1. **Variable name** This column specifies the variable name as provided in the original dataset. Variable names cannot be changed in *sdApp*, as they are unique identifiers. If the anonymization process renders a variable name no longer appropriate, the variable must be renamed after exporting the dataset in a software of choice.
2. **Type** Each variable has a internal *R* type. The different types include numeric, integer, factor and string. Each of the different functions in the SDC process requires a specific variable type, e.g., the weight needs to be numeric. If a variable is not of the appropriate type, the type of the variable needs to be changed before a selection is made (see the Section [Convert variable type](#)).
3. **Key variables** Variables that are determined as key variables in the disclosure scenario need to be selected here. By default the radiobutton is at *No*. A variable can either be a categorical key variable (*cat.*) or a numeric key variable (*cont.*). The sets of categorical key variables and numeric key variables are treated independently in *sdApp*. Categorical key variables can be of type integer or factor. Numeric key variables can be of type integer or numeric. At least one variable needs to be selected as categorcial key variable in order to create an SDC problem.
4. **Weight** The sampling weight is used to measure the disclosure risk. The weight variable needs to be of type numeric.
5. **Hierarchical identifier** If the data has a hierarchical structure, e.g., individuals in households, the variable that defines this hierarchy needs to be selected as hierarchical identifier (see also the Section *Risk*). This could be for instance a household ID. The hierarchical identifier needs to be a unique ID in the complete dataset and the same for each member of the hierarchical unit (household). If the unique hierachical indentifier is composed of several variabls, e.g., a geographical identifier, such as region, and a household ID which is unique within regions but not across, a unique hierarchical identifier needs to generated before importing

the data into *sdApp*. This can be done in a software of choice by concatenating the different components. The household identifier can be of type ...

6. **PRAM** If some variables are considered for application of the PRAM method (see [PRAM](#)), they need to be specified at this stage. PRAM variables can be of type ...
7. **Delete** Variables that need to be deleted from the dataset for release, such as direct identifiers, need to be selected here. Variables to be deleted can be of any type.
8. **Number of levels** This column shows the number of unique values in each variable. For instance a gender variable has typically two different levels. Note that if a variable contains missing values, this is also considered as a distinct value.
9. **Number of missing** This column indicates the number of missing values in each variable. If values were set to NA, the missing value code in R, these are counted here. Other missing value codes, such as 9, 99, 998 need to be set to NA (see also the Section [Set missing values to NA](#)).

Note: All variables need to be of the appropriate variable type. If the variable type of a variable is not suitable for the selected variable function, a popup window with an error message will appear. If necessary, the variable type needs to be changed before setting up the SDC problem.

Select variables i

Variable name	Type	Key variables			Weight	Hierarchical identifier	PRAM	Delete	Number of levels	Number of missing
urbrur	factor	<input type="radio"/> No	<input checked="" type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	0
roof	factor	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	5	0
walls	factor	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	3	0
water	integer	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8	0
electcon	integer	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3	0
relat	integer	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	9	0
sex	factor	<input type="radio"/> No	<input checked="" type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	0
age	factor	<input type="radio"/> No	<input checked="" type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	88	0
hhcivil	integer	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4	0
expend	integer	<input type="radio"/> No	<input type="radio"/> Cat.	<input checked="" type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4580	0
income	numeric	<input type="radio"/> No	<input type="radio"/> Cat.	<input checked="" type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1346	0
savings	numeric	<input type="radio"/> No	<input type="radio"/> Cat.	<input checked="" type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4579	0
ori_hid	integer	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000	0
sampling_weight	integer	<input checked="" type="radio"/> No	<input type="radio"/> Cat.	<input type="radio"/> Cont.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	0

Setup SDC problem

Fig. 5.1: Table on Anonymize tab for variable selection

Table 5.1: Columns in setup table

Column header	Description
Variable name	Name of variable in original dataset
Type	Variable type in R (factor, integer, numeric, character)
Key variables	Radio buttons to select variable as cat. or cont. key variable
Weight	Column to select variable as weight variable
Hierarchical identifier	Column to select variable as hierarchical identifier
PRAM	Column to select variable for PRAM method
Delete	Column to select variable to be deleted from released dataset
Number of levels	Number of different values (including NA/missing) in a categorical (type factor) variable
Number of missing	Number of records with missing value for this particular variable

Once a valid variable selection is made, a blue button will appear at the bottom of the setup table:

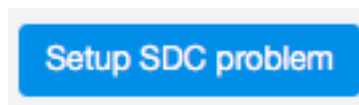


Fig. 5.2: Blue setup button appears below the setup table if the variable selection is valid

If a variable selection is invalid, the setup button will disappear and only reappears once all invalid choices are corrected. Pop-up windows as shown in Fig. 5.3, will guide the user through the variables that need to be fixed. The most common invalid choices are the selection of more than one function for a variable and the selection of a function that does not correspond with the variable type.

Before clicking the blue button to setup the SDC problem, several parameters have to be set, as outlined in the next section.

Note: If an invalid variable choice is made, such as an invalid variable type or a variable is selected for more than one choice, a pop-up window with an informative error message is shown. An example is shown in Fig. 5.3. The error message can be closed by clicking *Continue*. It is important to undo the invalid selection after clicking away the error message, as this doesn't happen automatically. Not correcting the selection will make it later difficult to trace back the invalid selections. The blue setup button disappears and reappears once the problem is fixed.

5.2 Settings

Besides the variable selection, there are two more parameters to be set before creating the SDC problem: alpha and seed. Both parameters can be set with sliders in the right sidepanel (see Fig. 5.4).

5.2.1 Alpha

The parameter alpha is used to compute the frequencies of keys, which is used to compute risk measures for categorical key variables. Alpha is the weight with which a key that coincides based on a missing value (NA) contributes to these frequencies. The default value of the parameter alpha is 1, which means that two records that have the same key (combination of values in key variables), are considered to coincide completely.

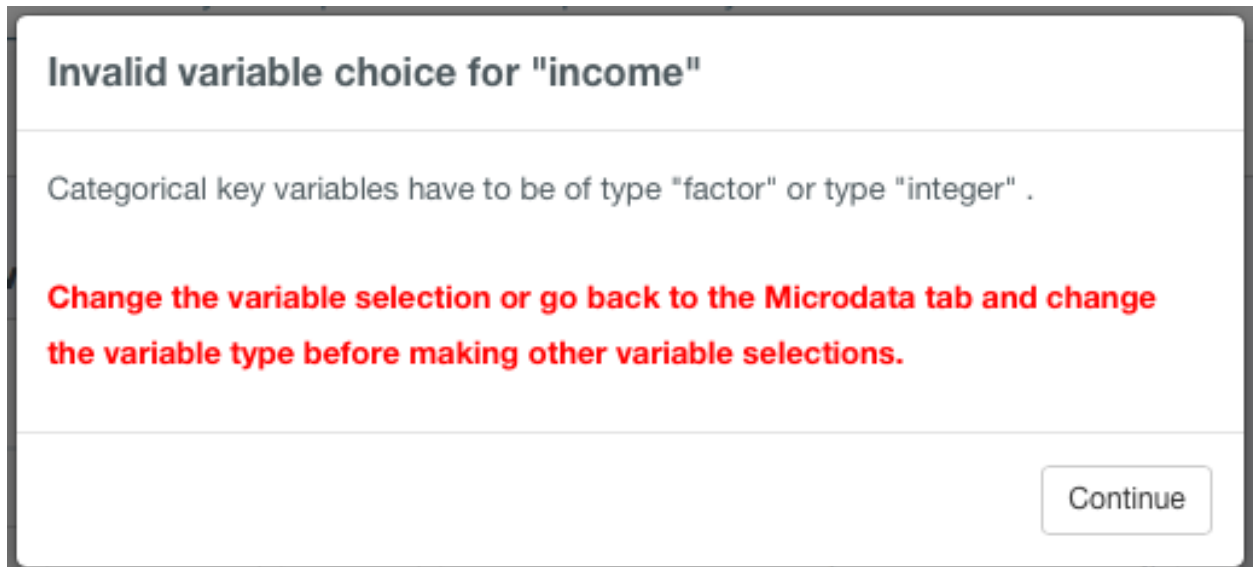


Fig. 5.3: Example of a popup window with an error message after an invalid variable choice



Fig. 5.4: Sliders to set additional parameters for the SDC problem

5.2.2 Seed

Every time a probabilistic method is used, a different outcome is generated. For these methods it is often recommended that a seed be set for the random number generator if you want to produce replicable results. The seed is used to initialize the random number generator used for probabilistic methods. In *sdcApp*, the seed can be set to any integer value from 0 to 500. To select a value, you can click with the mouse pointer on the slider and use the arrow keys (left and right or up and down) to select an exact value. In [Fig. 5.4](#) the seed is set at 388.

Note: In order to replicate exact results when using probabilistic methods, the order in which the methods are carried out influences the results. Therefore, besides the seed, also the order of the operations needs to be the same. The seed changes when used in the random number generator. When the undo button is used (see [Fig. 5.4](#)), the seed is not reset to the value prior to the reverted step.

5.3 Summary view

After setting up the SDC problem, the application jumps automatically to the summary view of the *Anonymize* tab. When an SDC problem is available, the *Anonymize* tab provides a summary of the SDC problem and allows to apply anonymization methods.

This tab first shows a Summary overview of the problem. The content of the summary page varies with the SDC problem. For example, if no numerical key variables were selected, the information on numeric key variables is omitted. [Fig. 5.3](#) shows the summary page.

RISK MEASUREMENT

6.1 Summary view

6.1.1 Global risk measure

For categorical variables

6.1.2 k -anonymity

The risk measure k -anonymity is based on the principle that, in a safe dataset, the number of individuals sharing the same combination of values (keys) of categorical quasi-identifiers should be higher than a specified threshold k . k -anonymity is a risk measure based on the microdata to be released, since it only takes the sample into account. An individual violates k -anonymity if the sample frequency count f_k for the key k is smaller than the specified threshold k . For example, if an individual has the same combination of quasi-identifiers as two other individuals in the sample, these individuals satisfy 3-anonymity but violate 4-anonymity. In the dataset, six individuals satisfy 2-anonymity and four violate 2-anonymity. The individuals that violate 2-anonymity are sample uniques. The risk measure is the number of observations that violates k -anonymity for a certain value of k , which is

$$\sum_i I(f_k < k),$$

where I is the indicator function and i refers to the i^{th} record. This is simply a count of the number of individuals with a sample frequency of their key lower than k . The count is higher for larger k , since if a record satisfies k -anonymity, it also satisfies $(k + 1)$ -anonymity. The risk measure k -anonymity does not consider the sample weights, but it is important to consider the sample weights when determining the required level of k -anonymity. If the sample weights are large, one individual in the dataset represents more individuals in the target population, the probability of a correct match is smaller, and hence the required threshold can be lower. Large sample weights go together with smaller datasets. In a smaller dataset, the probability to find another record with the same key is smaller than in a larger dataset. This probability is related to the number of records in the population with a particular key through the sample weights.

In the summary view

6.1.3 Risk measures for numerical key variables

6.1.4 Household risk

If household identifier is selected, household risk will automatically be displayed.

Information on k -anonymity

Below the number of observations violating k -anonymity is shown for the original data and the modified dataset

k -anonymity	Modified data	Original data
2-anonymity	38 (0.830%)	38 (0.830%)
3-anonymity	90 (1.965%)	90 (1.965%)
5-anonymity	239 (5.218%)	239 (5.218%)

Fig. 6.1: Information on k -anonymity violators in summary view

6.2 Detailed view

The Risk/Utility tab provides more detailed information on risk measures and records at (high) risk.

6.2.1 Risky observations

6.2.2 SUDA

The SUDA algorithm identifies all the MSUs in the sample, which in turn are used to assign a SUDA score to each record. This score indicates how “risky” a record is. The potential risk of the records is determined based on two observations:

- The smaller the size of the MSU within a record (i.e., the fewer variables are needed to reach uniqueness), the greater the risk of the record
- The larger the number of MSUs possessed by a record, the greater the risk of the record

The screenshot shows the sdcMicro GUI with the 'Risk/Utility' tab selected. The main area displays the 'SUDA2 risk measure' section, which includes a description of the algorithm and a slider to specify the sampling fraction for stratified sampling. The slider is set to 0.11. Below the slider is a 'Calculate suda2-scores' button. To the right, there is a 'Variable selection' table with columns for 'Variable name', 'Type', and 'Additional suppressions by local suppression algorithm'.

Variable name	Type	Additional suppressions by local suppression algorithm
urbrur	cat. key variable	0
sex	cat. key variable	0
age	cat. key variable	0
expend	num. key variable	

Fig. 6.2: Compute SUDA scores

6.2.3 l -diversity

A dataset satisfies l -diversity if for every key k there are at least l different values for each of the sensitive variables. In the example, the first two individuals satisfy only 1-diversity, individuals 4 and 6 satisfy 2-diversity. The required level of l -diversity depends on the number of possible values the sensitive variable can take. If the sensitive variable

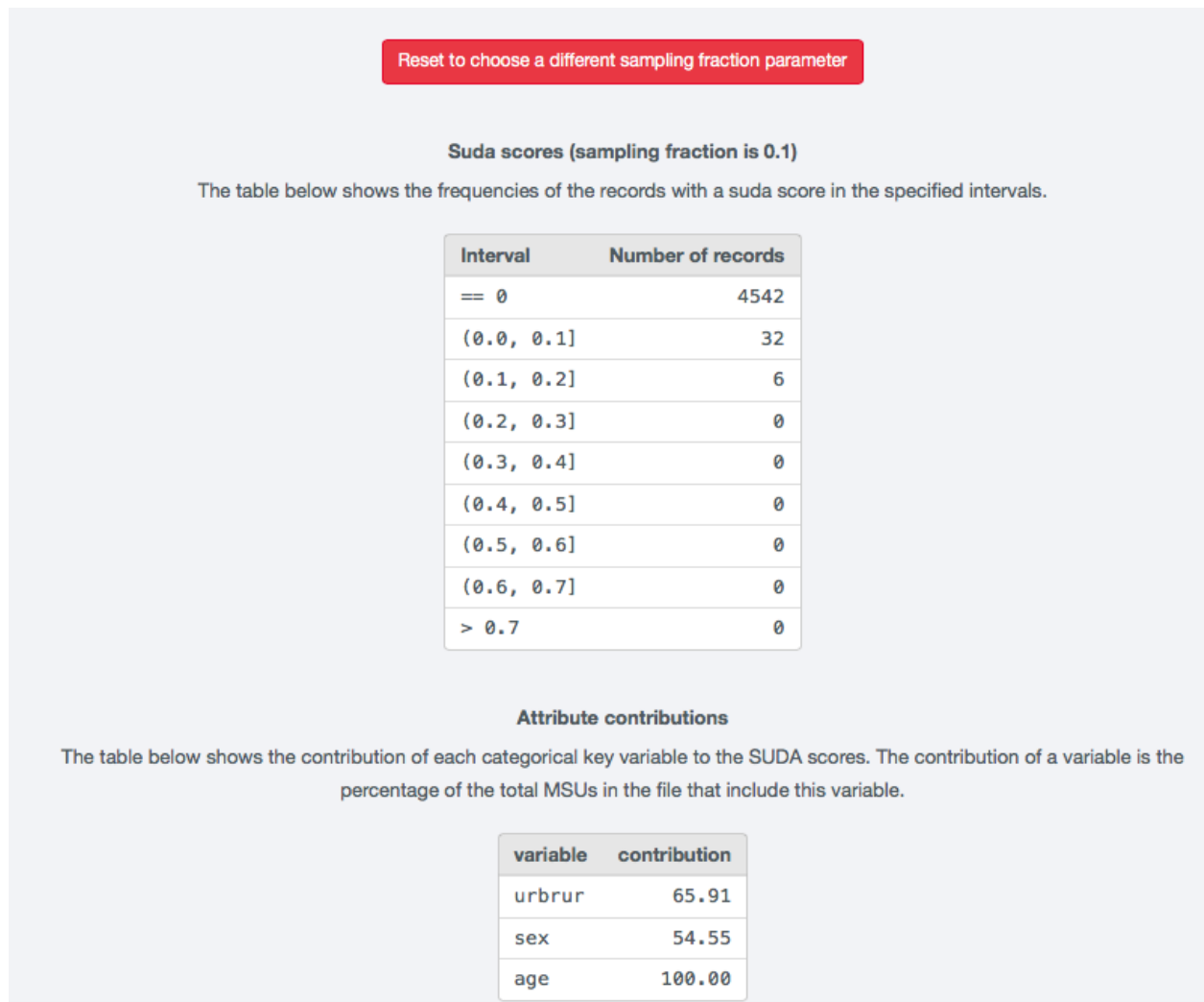


Fig. 6.3: Result of SUDA calculation

is a binary variable, the highest level of l -diversity that can be achieved is 2. A sample unique will always only satisfy 1-diversity.

To compute l -diversity for sensitive variables in sdcApp

6.2.4 k -anonymity

Information on k -anonymity

Below the number of observations violating k -anonymity is shown for the original data and the modified dataset

k -anonymity	Modified data	Original data
2-anonymity	38 (0.830%)	38 (0.830%)
3-anonymity	90 (1.965%)	90 (1.965%)
5-anonymity	239 (5.218%)	239 (5.218%)

Fig. 6.4: Information on k -anonymity violators for any level of k

ANONYMIZATION METHODS

Once the disclosure risk is evaluated and is too high for release, SDC methods need to be applied to the variables to reduce the risk. This process is iterative, i.e., after applying a certain method with a set of parameters, the disclosure risk needs to be reassessed and the information loss needs to be evaluated. If the result is not satisfactory, other methods can be applied to other variables. It is also possible to undo the method and reapply the same method with a different set of parameters.

In this section, we provide a brief description of common SDC methods for microdata and show how to use these in *sdcApp*. For more information on the choice of the appropriate method and more detailed information on the methods themselves, we refer to ...

7.1 Recoding

7.1.1 Global recoding

Global recoding combines several categories of a categorical variable or constructs intervals for continuous variables. This reduces the number of categories available in the data and potentially the disclosure risk, especially for categories with few observations, but also, importantly, it reduces the level of detail of information available to the analyst.

Variable needs to be of type factor and key variable

Variables can already be recoded before on data tab

Choose factor variable

age

Select levels to recode/combine

95 (1 obs) 90 (2 obs) 88 (1 obs)
85 (1 obs)

Specify new label for recoded values

85+

Add missing values to new factor level?

☒ no ☐ yes

Recode key variable

Fig. 7.1: Settings for global recoding to recode the variable age

7.1.2 Top and bottom coding

Top and bottom coding are similar to global recoding, but instead of recoding all values, only the top and/or bottom values of the distribution or categories are recoded. This can be applied only to ordinal categorical variables and (semi-)continuous variables, since the values have to be at least ordered. Top and bottom coding is especially useful if the bulk of the values lies in the center of the distribution with the peripheral categories having only few observations (outliers). Examples are age and income; for these variables, there will often be only a few observations above certain thresholds, typically at the tails of the distribution. The fewer the observations within a category, the higher the identification risk. One solution could be grouping the values at the tails of the distribution into one category. This reduces the risk for those observations, and, importantly, does so without reducing the data utility for the other observations in the distribution.

The screenshot shows the sdcMicro GUI interface for topcoding. It features a 'Select variable' dropdown menu with 'income' selected. Below it is a 'Set threshold value' input field containing '8000000'. To the right, there is a section 'Apply top/bottom coding?' with radio buttons for 'top' (selected) and 'bottom'. Below this is a 'Set replacement value' input field containing '83042423'. A summary line in red text states: '915 (out of 4580) values will be replaced! This equals 19.98 percent of the data.' At the bottom of this section is a blue button labeled 'Apply Top/Bottom-Coding'.

Fig. 7.2: Settings for topcoding the variable income at 8 million

Note: Top and bottom coding can only be applied to numeric variables. If age, as in our example, is converted to factor, the global recoding method needs to be used, in order to topcode age by grouping all values above the threshold.

Note: It is advised to use a replacement value different than the threshold value, such as the weighted mean or median to reduce information loss. The replacement value needs to be computed in a different software and manually inserted in *sdcApp*.

7.2 k-Anonymity / local suppression

It is common in surveys to encounter values for certain variables or combinations of quasi-identifiers (keys) that are shared by very few individuals. When this occurs, the risk of re-identification for those respondents is higher than the rest of the respondents (see the Section k-anonymity). Often local suppression is used after reducing the number of keys in the data by recoding the appropriate variables. Recoding reduces the number of necessary suppressions as well as the computation time needed for suppression. Suppression of values means that values of a variable are replaced by a missing value (NA in R). The the Section k-anonymity discusses how missing values influence frequency counts and k-anonymity.

7.2.1 Importance

7.2.2 Subsets

Do you want to apply the method for each group defined by the selected variable? ⓘ

no stratification

Do you want to modify importance of key variables for suppression? ⓘ

☒ No ☐ Yes

Tip - The total number of suppressions is likely to increase by specifying an importance vector. Specifying an importance vector can affect the computation time.

Apply k-anonymity to subsets of key variables? ⓘ

☒ No ☐ Yes

Set the k-anonymity parameter ⓘ

2 3 50

Establish k-anonymity

Fig. 7.3: Settings for local suppression to achieve 3-anonymity

Do you want to modify importance of key variables for suppression? ⓘ

☐ No ☒ Yes

Tip - The total number of suppressions is likely to increase by specifying an importance vector. Specifying an importance vector can affect the computation time.

Select the importance for key variable "urbrur"

1

Select the importance for key variable "sex"

2

Select the importance for key variable "age"

3

Fig. 7.4: Importance settings for local suppression

Apply k-anonymity to subsets of key variables? ⓘ

☐ No ☒ Yes

Apply k-anonymity to all subsets of 1 key variables?

☒ No ☐ Yes

Set k-anonymity parameter for combinations of 1 variables

Apply k-anonymity to all subsets of 2 key variables?

☒ No ☐ Yes

Set k-anonymity parameter for combinations of 2 variables

Apply k-anonymity to all subsets of 3 key variables?

☒ No ☐ Yes

Set k-anonymity parameter for combinations of 3 variables

Fig. 7.5: Subset settings for local suppression

7.3 PRAM

Select variable(s) for PRAM ⓘ

Postrandomize within different groups (stratification)? ⓘ

Choose value for 'pd' ⓘ

Choose value for 'alpha' ⓘ

Postrandomize

Fig. 7.6: Settings for PRAM

The PRAM algorithm randomly changes the values of selected variables in some records according to a custom-defined transition matrix.

The user can freely specify a transition matrix, which will be used for the post-randomization of a single variable. The requirement is that all row sums of the specified matrix sum up to 100!

Select variable for PRAM ⓘ

walls

Postrandomize within different groups (stratification)? ⓘ

no stratification

Specify the transition matrix. Note: the entries in each rows must add up to 100. ⓘ

2	3	9
50	20	30
10	70	20
0	0	100

Postrandomize

Fig. 7.7: Settings for PRAM with customized transition matrix

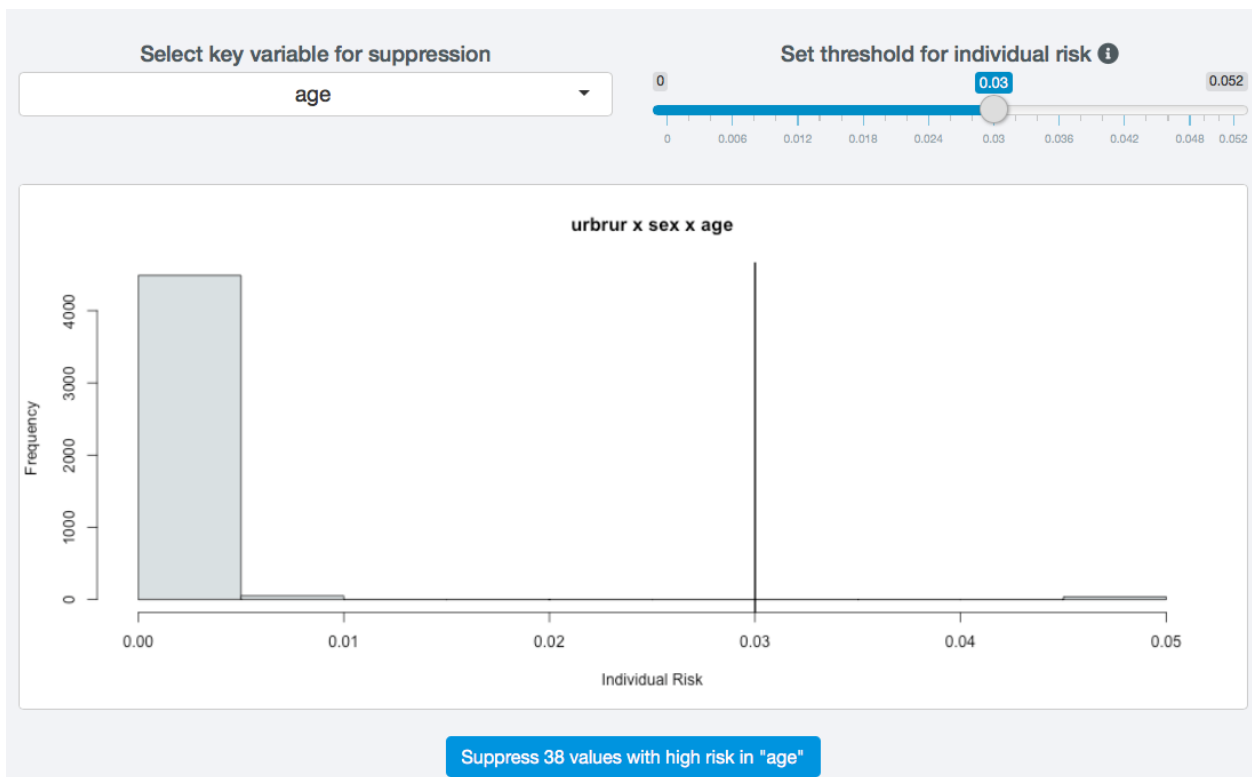


Fig. 7.8: Settings for suppressing values in records with high risk

Clustermethod ⓘ

☒ clara
☐ pam
☐ kmeans
☐ cmeans
☐ bclust

Transformation ⓘ

☒ none
☐ log
☐ boxcox

Number of clusters ⓘ

3

Fig. 7.9: Settings for microaggregation

Fig. 7.10: Additional settings for microaggregation

Fig. 7.11: Cluster settings for microaggregation

7.4 Suppress values with high risk

7.5 Top/Bottom coding

7.6 Microaggregation

7.7 Adding noise

7.8 Rank swapping

7.9 Undo

Finding an anonymization strategy for a microdata dataset is a trial-and-error process. The effect on risk and utility of different methods with different parameter settings can only be assessed by executing the methods on the actual dataset. Therefore, it is unlikely to find a satisfactory anonymization strategy at the first attempt. Before another method is applied, the previous method needs to be canceled. In *sdcApp* it is possible to undo the last method applied with one click. To test the effect of a combination of several methods, which is recommended, it is necessary to cancel several steps. To do so, the state of the SDC problem before applying the methods is saved to disk and can be reloaded afterwards. This is the same as canceling several steps. Both methods are described below.

7.9.1 Undo one step

In order to undo one step,
risk measures etc are also reset, script not, random seed not

7.9.2 Undo several steps

Recommended to
Save and reload

UTILITY MEASUREMENT

8.1 General utility measures in *sdApp*

8.1.1 Compare summary statistics

Categorical variables

Continuous variables

8.1.2 IL1s measure

8.2 Customized utility measures

As the statistical analyses based on the microdata depend, amongst others, on to the topic of the survey, the country and the definition of the variables, it is not feasible to include all these measures in *sdApp*. Instead, it is recommended to compute the statistics and indicators and perform statistical and econometric analyses on the original and anonymized datasets and evaluate the differences in the results. If a publication based on the microdata is already published, it is recommended to recompute the statistics in these publications from the anonymized dataset.

The approach is to compare the indicators calculated on the untreated data and the data after anonymization with different methods. If the differences between the indicators are not too large, the anonymized dataset can be released for use by researchers. It should be taken into account that indicators calculated on samples are estimates with a certain variance and confidence interval. Therefore, for sample data, it is more informative to compare the overlap of confidence intervals and/or to evaluate whether the point estimate calculated after anonymization is contained within the confidence interval of the original estimate.

Note: Some analyses may no longer be possible or not possible in exactly the same way. E.g. regression on age if age is recoded in 5-year intervals.

In order to do so, it is possible to export the dataset at any point in *sdApp*. See Export dataset. Several datasets can be exported after applying different methods with different parameters settings to compare the information loss resulting from the anonymization. This information can be used to select the anonymization methods as well as to inform the user about the implications of the anonymization on the validity of the dataset for analysis.

EXPORT DATA AND REPORTS

9.1 Export anonymized dataset

sdApp supports datasets in several foreign data formats. The file formats that are supported for loading microdata are also supported for export (cf. Table 9.1).

In order to export the file, click on **Anonymized Data** in the left sidebar on the **Export Data** tab. The dataset shown is the file as it will be exported. Select the appropriate file format with the radiobuttons underneath the data.

In case the microdata is exported as csv or STATA file, additional options need to be specified. For a csv file, whether first row should include column names, the field separator as well as the decimal separator. For STATA files, the version of STATA needs to be specified. STATA cannot STATA files saved for a higher version.

Option to randomize the order of the records. Order may reveal values, e.g. ordered by region with suppressed region value Need to replace existing ID

In order to export the dataset, click on blue button **Save dataset**. The dataset is saved to the , exported two with file name. . . Exported to the microdata, select

Table 9.1: Data formats compatible with *sdApp*

Software	File extension
R/RStudio	.rdata
SPSS	.sav
SAS	.sas7bdat
CSV	.csv, .txt
STATA	.dta

Also for intermediate export

..NOTE:: Categorical variables (type factor in *sdApp*) that were had a value and a label in the input dataset are labels (variable, value), coding 0,1 to 1,2 etc.

Extra for STATA input files: change variable labels for STATA files

9.2 Exporting reports

It is extremely important to document the SDC process of microdata both for internal use as well as for external use. The internal report should contain detailed descriptions of all steps carried out as well as reasoning for

9.2.1 Internal report

An important step in the SDC process is reporting, both internal and external. Internal reporting contains the exact description of anonymization methods used, parameters but also the risk measures before and after anonymization. This allows replication of the anonymized dataset and is important for supervisory authorities/bodies to ensure the anonymization process is sufficient to guarantee anonymity according to the applicable legislation.

Report is just technical overview, not complete

file path, name of file

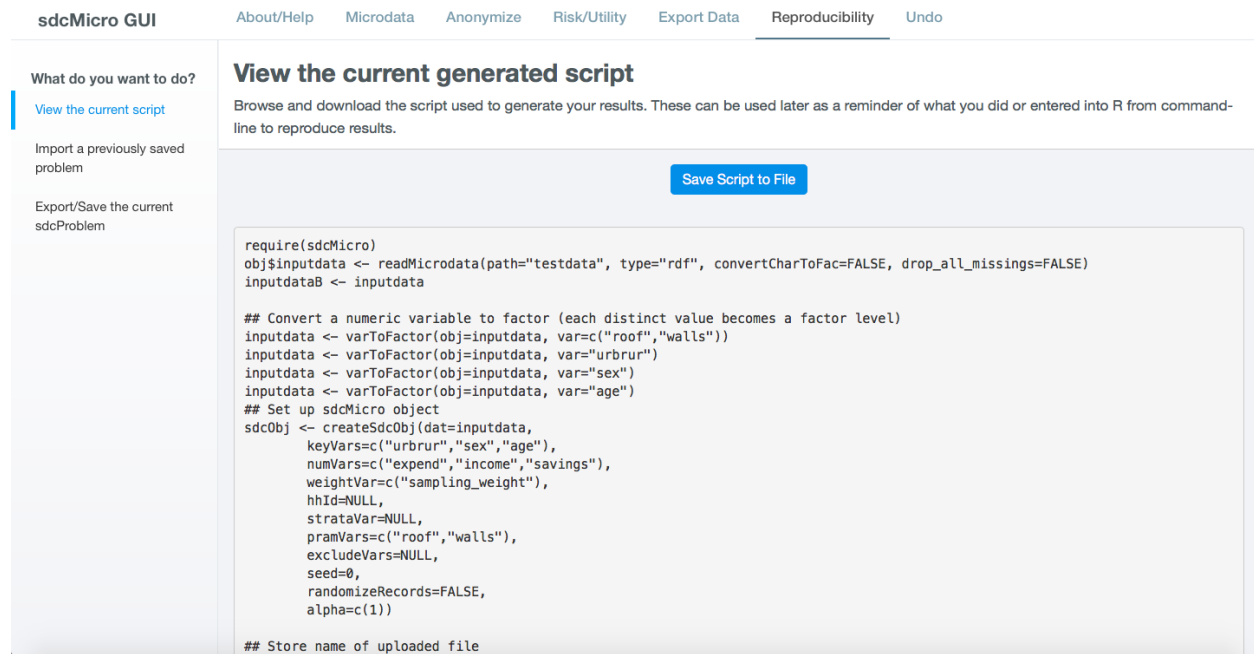


Fig. 9.1: Exporting an internal report

9.2.2 External report

The external report

External reporting informs the user that the data has been anonymized, provides information for valid analysis on the data and explains the limitations to the data as a result of the anonymization. A brief description of the methods used can be included. The release of anonymized microdata should be accompanied by the usual metadata of the survey (survey weight, strata, survey methodology) as well as information on the anonymization methods that allow researchers to do valid analysis (e.g., amount of noise added, transition matrix for PRAM).

file path, name of file

REPRODUCIBILITY

Reproducibility is key to the SDC process, as ...

10.1 Exporting *R* script

sdApp is a GUI for the *R* package *sdMicro*. All steps executed in *sdApp* are translated into *R* commands. Therefore, the full anonymization can also be performed from command-line in *R*. While carrying out the anonymization process, the code to perform the same action from command-line is generated. The code can be viewed and exported on the **Reproducibility** tab by selecting **View the current script** from the left sidebar (cf. Fig. 10.1). The script also contains comments, which are the lines starting with the hash tag (#). These comments are meant to help with the interpretation of the code blocks.

sdMicro GUI About/Help Microdata Anonymize Risk/Utility Export Data **Reproducibility** Undo

What do you want to do?

- [View the current script](#)
- Import a previously saved problem
- Export/Save the current sdcProblem

View the current generated script

Browse and download the script used to generate your results. These can be used later as a reminder of what you did or entered into R from command-line to reproduce results.

[Save Script to File](#)

```
require(sdMicro)
obj$inputdata <- readMicrodata(path="testdata", type="rdf", convertCharToFac=FALSE, drop_all_missings=FALSE)
inputdataB <- inputdata

## Convert a numeric variable to factor (each distinct value becomes a factor level)
inputdata <- varToFactor(obj=inputdata, var=c("roof","walls"))
inputdata <- varToFactor(obj=inputdata, var="urbrur")
inputdata <- varToFactor(obj=inputdata, var="sex")
inputdata <- varToFactor(obj=inputdata, var="age")
## Set up sdMicro object
sdcObj <- createSdcObj(dat=inputdata,
  keyVars=c("urbrur","sex","age"),
  numVars=c("expend","income","savings"),
  weightVar=c("sampling_weight"),
  hhId=NULL,
  strataVar=NULL,
  pramVars=c("roof","walls"),
  excludeVars=NULL,
  seed=0,
  randomizeRecords=FALSE,
  alpha=c(1))

## Store name of uploaded file
```

Fig. 10.1: *R* script to reproduce anonymization process after setting up SDC problem

The goal of the *R* script is threefold: 1) To reproduce the steps taken in the anonymization process. This guarantees the reproducibility, since the all variable selections and parameters are contained in the code and the order of the application of different methods is preserved in the code. 2) As a starting point to learn *R* and use *sdMicro* from *R* command-line. Especially for users with some degree of familiarity with *R*, the script 3) To rerun the same methods with different parameter settings without the need to make all selections by mouseclick in the GUI. It's relatively easy

to change the parameter settings in the *R* code and rerun the code. However, the code does not include commands to show the results.

By clicking on the blue button **Save script to file** at the top of the page, the script is saved as *R* script (extension .R) on disk to the selected storage path on the **About/Help** tab (see [Introduction to sdcApp](#)). The filename of the exported script starts with 'exportedScript_sdcMicro' followed by a date and time stamp, e.g., 'exportedScript_sdcMicro_20181010_1212.R'.

In order to run the script in *R*, open the saved script in *RStudio*. The only thing to do is to change the path of the input file to the actual file path on your computer. In the second line of the *R* script,

Note: In case a method was applied in *sdcApp* and subsequently reverted by using the **Undo** button, the method is not erased from the script, but rather the undo command is added. For example, if local suppression was applied and reverted, this appears as follows in the script:

```
1 ## Local suppression to obtain k-anonymity
2 sdcObj <- kAnon(sdcObj, importance=c(1,2,3), combs=NULL, k=c(3))
3 sdcObj <- undolast(sdcObj)
```

This code preceding the :code:'undoLast'command and the :code:'undoLast'command can be deleted without changing the results.

10.2 Exporting reports

It is extremely important to document the SDC process of microdata both for internal use as well as for external use. The internal report should contain detailed descriptions of all steps carried out as well as reasoning for

10.2.1 Internal report

An important step in the SDC process is reporting, both internal and external. Internal reporting contains the exact description of anonymization methods used, parameters but also the risk measures before and after anonymization. This allows replication of the anonymized dataset and is important for supervisory authorities/bodies to ensure the anonymization process is sufficient to guarantee anonymity according to the applicable legislation.

Report is just technical overview, not complete

10.2.2 External report

The external report

External reporting informs the user that the data has been anonymized, provides information for valid analysis on the data and explains the limitations to the data as a result of the anonymization. A brief description of the methods used can be included. The release of anonymized microdata should be accompanied by the usual metadata of the survey (survey weight, strata, survey methodology) as well as information on the anonymization methods that allow researchers to do valid analysis (e.g., amount of noise added, transition matrix for PRAM).

sdcMicro GUI
About/Help
Microdata
Anonymize
Risk/Utility
Export Data
Reproducibility
Undo

What do you want to do?

[View the current script](#)

Import a previously saved problem

Export/Save the current sdcProblem

View the current generated script

Browse and download the script used to generate your results. These can be used later as a reminder of what you did or entered into R from command-line to reproduce results.

[Save Script to File](#)

```

require(sdcMicro)
obj$inputdata <- readMicrodata(path="testdata", type="rdf", convertCharToFac=FALSE, drop_all_missings=FALSE)
inputdataB <- inputdata

## Convert a numeric variable to factor (each distinct value becomes a factor level)
inputdata <- varToFactor(obj=inputdata, var=c("roof","walls"))
inputdata <- varToFactor(obj=inputdata, var="urbrur")
inputdata <- varToFactor(obj=inputdata, var="sex")
inputdata <- varToFactor(obj=inputdata, var="age")
## Set up sdcMicro object
sdcObj <- createSdcObj(dat=inputdata,
  keyVars=c("urbrur","sex","age"),
  numVars=c("expend","income","savings"),
  weightVar=c("sampling_weight"),
  hhId=NULL,
  strataVar=NULL,
  pramVars=c("roof","walls"),
  excludeVars=NULL,
  seed=0,
  randomizeRecords=FALSE,
  alpha=c(1))

## Store name of uploaded file

```

Fig. 10.2: Exporting an internal report

UNDO

Microdata anonymization is a trial-and-error process. It is necessary to several methods, each with different parameter settings, to find optimal set of anonymization measures that minimize the information loss, while reducing the risk of disclosure to an acceptable level. Before applying an alternative method to the same variable or set of variables, it is important to undo the previously applied methods. Only in this way, it is possible to compare the effect on risk and information loss of a particular method or parameter setting. For instance to compare the effect of recoding the age variable in 5 or 10 year intervals, it is necessary to first undo the recoding in 5-year intervals before recoding in 10 year intervals.

In *sdcApp* it is possible to undo the last anonymization step. In order to undo several steps, it is required to save and reload the SDC problem. Both ways are explained below.

11.1 Undo one step

In order to undo one step, go to the **Undo** tab. The screen shows risk measures etc are also reset, script not (completely), random seed not

11.2 Undo several steps

Save and reload

Recommended to save the SDC problem after each method to be able to reload. This is also practical if *sdcApp* or *R* crash. Also useful to continue working at a later point of transfer a problem to a different computer

11.2.1 Save a previously saved problem

11.2.2 Load a previously saved problem

..NOTE:: Different file names for different files (data, sdcProblem)