Linking cell suppression and the software R
A primer on R-package sdcTable

Bernhard Meindl (bernhard.meindl@statistik.gv.at)

Statistics Austria

NTTS 2009, Brussels
Overview

1. Another software for tabular data protection

2. The software used

3. Overview of sdcTable

4. Strengths and Limitations

5. Example: Protection tabular data using sdcTable
Data in official statistics is often disseminated in tabular data.
Data in official statistics is often disseminated in tabular data.

NSI’s need to protect data in order to:

- satisfy legal demands
- the respondents’ right for confidentiality
Protection of tabular data

- Data in official statistics is often disseminated in tabular data.
- NSI’s need to protect data in order to:
  - satisfy legal demands
  - the respondents’ right for confidentiality
Protection of tabular data

- Data in official statistics is often disseminated in tabular data.
- NSI’s need to protect data in order to:
  - satisfy legal demands
  - the respondents’ right for confidentiality
- Popular methods for protecting tabular data include:
Protection of tabular data

- Data in official statistics is often disseminated in tabular data.
- NSI’s need to protect data in order to:
  - satisfy legal demands
  - the respondents’ right for confidentiality
- Popular methods for protecting tabular data include:
  - Cell suppression
  - Rounding methods (simple, random, controlled)
  - Controlled Tabular Adjustment (CTA)
Another software for tabular data protection

Status quo

- The standard software for tabular data protection is $\tau$-Argus.
Another software for tabular data protection

Status quo

- The standard software for tabular data protection is $\tau$-Argus.
- Unfortunately, the source code of $\tau$-Argus is not available under a public license.
Another software for tabular data protection

Status quo

- The standard software for tabular data protection is $\tau$-Argus.
- Unfortunately, the source code of $\tau$-Argus is not available under a public license.
- It is therefore not possible to . . .:
The standard software for tabular data protection is \( \tau \)-Argus.

Unfortunately, the source code of \( \tau \)-Argus is not available under a public license.

It is therefore not possible to . . . :

- improve the software.
- modify the software.
- learn from the existing code.
Another software for tabular data protection

Status quo

- The standard software for tabular data protection is $\tau$-Argus.
- Unfortunately, the source code of $\tau$-Argus is not available under a public license.
- It is therefore not possible to . . . :
  - improve the software.
  - modify the software.
  - learn from the existing code.
- For real world problems, it is often necessary to adjust software in order to make things work.
Intentions for developing a new software

- A new software for tabular data should be developed in an open way.
Intentions for developing a new software

- A new software for tabular data should be developed in an open way.
- Everybody should be able to learn from the code.
Intentions for developing a new software

- A new software for tabular data should be developed in an open way.
- Everybody should be able to learn from the code.
- It should be possible to make suggestions for improvements.
Intentions for developing a new software

- A new software for tabular data should be developed in an **open** way.
- Everybody should be able to learn from the code.
- It should be possible to make suggestions for improvements.
- The software should be flexible and extensible.
Intentions for developing a new software

- A new software for tabular data should be developed in an open way.
- Everybody should be able to learn from the code.
- It should be possible to make suggestions for improvements.
- The software should be flexible and extensible.

→ Idea: linking disclosure control for tabular data with the power of R.
Why R?

- R is free software for statistical computing and graphics.
Why R?

- R is free software for statistical computing and graphics.
- R runs on all major platforms (Linux, Windows, Mac).
Why R?

- R is free software for statistical computing and graphics.
- R runs on all major platforms (Linux, Windows, Mac).
- R can easily be extended using packages.
Why R?

- R is free software for statistical computing and graphics.
- R runs on all major platforms (Linux, Windows, Mac).
- R can easily be extended using packages.
- Additional packages can use methods/functions from other packages.
Why R?

- R is free software for statistical computing and graphics.
- R runs on all major platforms (Linux, Windows, Mac).
- R can easily be extended using packages.
- Additional packages can use methods/functions from other packages.

→ **Idea:** Extending R with a package on SDC for tabular data.
Design issues - key questions

- how should tabular data be organized?
Design issues - key questions

- how should tabular data be organized?
  - R is object oriented language.
  - we can define that input objects need to be objects of specific classes.
  - objects belonging to specific classes need to be built specifically.
Overview of sdcTable

Design issues - key questions

- how should tabular data be organized?
  - R is object oriented language.
  - we can define that input objects need to be objects of specific classes.
  - objects belonging to specific classes need to be built specifically.
- which algorithms should be implemented?
  - algorithms for secondary cell suppression problem.
  - rounding procedures.
  - controlled tabular adjustment.
  - Functions for comparing and/or summarizing results should be available.
Design issues - key questions

- how should tabular data be organized?
  - R is object oriented language.
  - we can define that input objects need to be objects of specific classes.
  - objects belonging to specific classes need to be built specifically.

- which algorithms should be implemented?
  - algorithms for secondary cell suppression problem.
  - rounding procedures.
  - controlled tabular adjustment.
Design issues - key questions

- how should tabular data be organized?
  - R is object oriented language.
  - we can define that input objects need to be objects of specific classes.
  - objects belonging to specific classes need to be built specifically.

- which algorithms should be implemented?
  - algorithms for secondary cell suppression problem.
  - rounding procedures.
  - controlled tabular adjustment.

- Functions for comparing and/or summarizing results should be available.
• a first version of sdcTable is available on CRAN.
current state

- a first version of sdcTable is available on CRAN.
- algorithms for secondary cell-suppression problem are implemented:
current state

- A first version of `sdcTable` is available on CRAN.
- Algorithms for secondary cell-suppression problem are implemented:
  - HiTaS: A heuristic approach to cell suppression in hierarchical tables.
  - GHMITER: Procedure to search for and suppress hypercubes.
  - Simple rounding.
  - Random rounding.
  - Controlled rounding.
  - Draft implementation of controlled tabular adjustment (CTA).

Note: the implementation needs more testing.
current state

- A first version of sdcTable is available on CRAN.
- Algorithms for secondary cell-suppression problem are implemented:
  - HiTaS: A heuristic approach to cell suppression in hierarchical tables.
  - GHMITER: Procedure to search for and suppress hypercubes.
- HiTaS and GHMITER are heuristic algorithms.
current state

- a first version of sdcTable is available on CRAN.
- algorithms for secondary cell-suppression problem are implemented:
  - HiTaS: A heuristic approach to cell suppression in hierarchical tables.
  - GHMITER: procedure to search for and suppress hypercubes.

- HiTaS and GHMITER are heuristic algorithms.
- implemented algorithms for rounding of tabular data:
a first version of sdcTable is available on CRAN.

algorithms for secondary cell-suppression problem are implemented:

- **HiTaS**: A heuristic approach to cell suppression in hierarchical tables.
- **GHMITER**: procedure to search for and suppress hypercubes.

**HiTaS** and **GHMITER** are heuristic algorithms.

implemented algorithms for rounding of tabular data:

- simple rounding.
- random rounding.
- controlled rounding.

Note: the implementation needs more testing.
a first version of sdcTable is available on CRAN.
algorithms for secondary cell-suppression problem are implemented:
  - HiTaS: A heuristic approach to cell suppression in hierarchical tables.
  - GHMITER: procedure to search for and suppress hypercubes.
HiTaS and GHMITER are heuristic algorithms.
implemented algorithms for rounding of tabular data:
  - simple rounding.
  - random rounding.
  - controlled rounding.
draft implementation of controlled tabular adjustment (CTA).
Overview of sdcTable

current state

- A first version of sdcTable is available on CRAN.
- Algorithms for secondary cell-suppression problem are implemented:
  - HiTaS: A heuristic approach to cell suppression in hierarchical tables.
  - GHMITER: Procedure to search for and suppress hypercubes.
- HiTaS and GHMITER are heuristic algorithms.
- Implemented algorithms for rounding of tabular data:
  - Simple rounding.
  - Random rounding.
  - Controlled rounding.
- Draft implementation of controlled tabular adjustment (CTA).
- Note: the implementation needs more testing.
Strengths

- The code is open source and can be adjusted and improved by everyone.
Strengths

- The code is open source and can be adjusted and improved by everyone.
- An implementation goal was to program sdcTable be as flexible as possible.
Strengths

- The code is open source and can be adjusted and improved by everyone.
- An implementation goal was to program sdcTable be as flexible as possible.
- Additional algorithms could be *plugged-in* (relatively easy) by re-using existing structures.
Strengths

- The code is open source and can be adjusted and improved by everyone.
- An implementation goal was to program `sdcTable` be as flexible as possible.
- Additional algorithms could be *plugged-in* (relatively easy) by re-using existing structures.
- The package can be easily extended and further developed.
Limitations

- sdcTable is still in early development.
Limitations

- `sdcTable` is still in early development.
- The implemented code still needs a lot of testing.
Limitations

- sdcTable is still in early development.
- the implemented code still needs a lot of testing.
- my (and possibly contributed) code is visible for everyone.
Limitations

- sdcTable is still in early development.
- the implemented code still needs a lot of testing.
- my (and possibly contributed) code is visible for everyone.
- Question: is this really a limitation (or even a weakness?)
Limitations

- `sdcTable` is still in early development.
- The implemented code still needs a lot of testing.
- My (and possibly contributed) code is visible for everyone.
- Question: is this really a limitation (or even a weakness?)
- Some algorithms need to be improved to work (automatically) for hierarchical tables.
Limitations

- `sdcTable` is still in early development.
- the implemented code still needs a lot of testing.
- my (and possibly contributed) code is visible for everyone.
- Question: is this really a limitation (or even a weakness?)
- some algorithms need to be improved to work (automatically) for hierarchical tables.
- if not mentioned yet: the implemented code still needs a lot of testing.
Example: Protection tabular data using \texttt{sdcTable}

The data

<table>
<thead>
<tr>
<th>SEX</th>
<th>REGION</th>
<th>VAL</th>
<th>SEX</th>
<th>REGION</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Total</td>
<td>160</td>
<td>Total</td>
<td>Main Region B</td>
<td>17</td>
</tr>
<tr>
<td>Male</td>
<td>Total</td>
<td>80</td>
<td>Male</td>
<td>Main Region B</td>
<td>5</td>
</tr>
<tr>
<td>Female</td>
<td>Total</td>
<td>80</td>
<td>Female</td>
<td>Main Region B</td>
<td>12</td>
</tr>
<tr>
<td>Total</td>
<td>Main Region A</td>
<td>124</td>
<td>Total</td>
<td>sub-Region B1</td>
<td>10</td>
</tr>
<tr>
<td>Male</td>
<td>Main Region A</td>
<td>64</td>
<td>Male</td>
<td>sub-Region B1</td>
<td>3</td>
</tr>
<tr>
<td>Female</td>
<td>Main Region A</td>
<td>60</td>
<td>Female</td>
<td>sub-Region B1</td>
<td>7</td>
</tr>
<tr>
<td>Total</td>
<td>sub-Region A1</td>
<td>11</td>
<td>Total</td>
<td>sub-Region B2</td>
<td>7</td>
</tr>
<tr>
<td>Male</td>
<td>sub-Region A1</td>
<td>4</td>
<td>Male</td>
<td>sub-Region B2</td>
<td>2</td>
</tr>
<tr>
<td>Female</td>
<td>sub-Region A1</td>
<td>7</td>
<td>Female</td>
<td>sub-Region B2</td>
<td>5</td>
</tr>
</tbody>
</table>

\ldots

| Total | sub-Region A10 | 19  |
| Male  | sub-Region A10 | 11  |
| Female| sub-Region A10 | 8   |

\textbf{Table:} (parts) of the dataset to protect.
SEX and REGION are the variables that define the table.
Standardisation of dimensional variables

- **SEX** and **REGION** are the variables that define the table.
- both **SEX** and **REGION** are hierarchical variables.
SEX and REGION are the variables that define the table.
both SEX and REGION are hierarchical variables.
Variable SEX consists of 2 levels.
SEX and REGION are the variables that define the table.

both SEX and REGION are hierarchical variables.

Variable SEX consists of 2 levels.

Variable REGION consists of 3 levels.
SEX and REGION are the variables that define the table.
both SEX and REGION are hierarchical variables.
Variable SEX consists of 2 levels.
Variable REGION consists of 3 levels.

→ Standardisation:
it is necessary to standardize characteristics of hierarchical variables to simplify further processing.
How to standardize dimensional variables

- Variable **REGION** may be represented as:
How to standardize dimensional variables

- Variable **REGION** may be represented as:
How to standardize dimensional variables

- The *grand total* can be represented as ’0000’.
How to standardize dimensional variables

- The *grand total* can be represented as ’0000’.
How to standardize dimensional variables

- Standardizing the second-level characteristics.
How to standardize dimensional variables

- Standardizing the second-level characteristics.
How to standardize dimensional variables

- Standardizing the third-level characteristics (part 1).
How to standardize dimensional variables

- Standardizing the third-level characteristics (part 1).
How to standardize dimensional variables

- Standardizing the third-level characteristics (part 2).
How to standardize dimensional variables

- Standardizing the third-level characteristics (part 2).

```
Total (0000)
region A (0100)
  sub-region 01 (0101)
  sub-region 02 (0102)
  ...
  sub-region 10 (0110)
region B (0200)
  sub-region 01 (0201)
  sub-region 02 (0202)
```
Creating an object of class \textit{fullData}: 

- Dimensional variables must be recoded to standard-form.
Creating an object of class *fullData*:

- Dimensional variables must be recoded to standard-form.
- Creation of a minimal dataset containing:
Creating an object of class \texttt{fullData}:

- Dimensional variables must be recoded to standard-form.
- Creation of a minimal dataset containing:
  - all combinations of the dimensional variables excluding all (sub)totals
  - an extra column containing the cell values is mandatory (of course)
Creating an object of class *fullData*:

- Dimensional variables must be recoded to standard-form.
- Creation of a minimal dataset containing:
  - all combinations of the dimensional variables excluding all (sub)totals
  - an extra column containing the cell values is mandatory (of course)
- the position of the dimensional variables within the minimal data-set.
Creating an object of class `fullData`:

- Dimensional variables must be recoded to standard-form.
- Creation of a minimal dataset containing:
  - all combinations of the dimensional variables excluding all (sub)totals
  - an extra column containing the cell values is mandatory (of course)
- the position of the dimensional variables within the minimal data-set.
- the hierarchical structure of all dimensional variables.
Creating an object of class *fullData*:

- Dimensional variables must be recoded to standard-form.
- Creation of a minimal dataset containing:
  - all combinations of the dimensional variables excluding all (sub)totals
  - an extra column containing the cell values is mandatory (of course)
- the position of the dimensional variables within the minimal data-set.
- the hierarchical structure of all dimensional variables.
- using function `createFullData()` suitable input objects to be used by the protection procedures can be generated.
Creating an object of class `fullData`:

- Dimensional variables must be recoded to standard-form.
- Creation of a minimal dataset containing:
  - all combinations of the dimensional variables excluding all (sub)totals
  - an extra column containing the cell values is mandatory (of course)
- the position of the dimensional variables within the minimal data-set.
- the hierarchical structure of all dimensional variables.
- using function `createFullData()` suitable input objects to be used by the protection procedures can be generated.
- it is possible to demand (simple) primary suppression rules as well.
Example: Protection tabular data using sdcTable

Creating an object of class `fullData`:

```r
# install and load the package
install.packages('sdcTable'); library(sdcTable)

# minData: the dataset without (sub)totals (excerpt)
print(minDat[1:4,])

SEX REGION  VAL
1 01 0100 64
2 02 0100 60
3 01 0101  7
4 02 0101  4

# position of dimensional variables
indexDimVars <- c(1,2)
```
Creating an object of class \texttt{fullData}:

```r
# hierarchical structure
structDimVars <- list()

# SEX
structDimVars[[1]] <- c(1,1)

# REGION
structDimVars[[2]] <- c(1,1,2)

# creating the dataset needed for protection procedure
fullData <- createFullData (minData,
                           indexDimVars, 
                           structDimVars, 
                           suppVals=TRUE,
                           suppLimit=3)

class(fullData)
[1] "fullData"
```
Protection data:

- objects generated with function `createFullData()` are of class `fullData`.

```r
res1 <- protectTable(fullData, method = "HYPERCUBE");
summary(res1)
```

```r
res2 <- protectTable(fullData, method = "HITAS");
summary(res2)
```
Protection data:

- objects generated with function `createFullData()` are of class `fullData`.
- Such objects can be used as input objects for function `protectTable()`.

```r
res1 <- protectTable(fullData, method = "HYPERCUBE");
summary(res1)
```

```r
res2 <- protectTable(fullData, method = "HITAS");
summary(res2)
```
Protection data:

- objects generated with function `createFullData()` are of class `fullData`.
- Such objects can be used as input objects for function `protectTable()`.
- `protectTable()` protects the given input data according to the method chosen.
Protection data:

- objects generated with function `createFullData()` are of class `fullData`.
- Such objects can be used as input objects for function `protectTable()`.
- `protectTable()` protects the given input data according to the method chosen.

```r
# using method Hypercube
res1 <- protectTable(fullData, method="HYPERCUBE");
summary(res1)

# using Hitas-approach
res2 <- protectTable(fullData, method="HITAS");
summary(res2)
```
Comments

- Output objects of function `protectTable()` are of class `safeTable`. 
Comments

- Output objects of function `protectTable()` are of class `safeTable`.
- Output objects contain the protected statistical table.
Comments

- Output objects of function `protectTable()` are of class `safeTable`.
- Output objects contain the protected statistical table.
- A summary method is provided for objects of class `safeTable`.
Comments

- Output objects of function `protectTable()` are of class `safeTable`.
- Output objects contain the protected statistical table.
- A summary method is provided for objects of class `safeTable`.
- The summary method gives information on:
Output objects of function `protectTable()` are of class `safeTable`. Output objects contain the protected statistical table. A summary method is provided for objects of class `safeTable`. The summary method gives information on:

- the protection algorithm used.
- total running time.
- number of necessary runs through the table (method `HYPERCUBE`)
- number of primary suppressions.
- total number of additional suppressions.
Outline/Conclusions

- sdcTable is the first try to link statistical disclosure control and R.
Outline/Conclusions

- sdcTable is the first try to link statistical disclosure control and R.
- sdcTable is free and open source and can be downloaded from CRAN (cran.r-project.org)
sdcTable is the first try to link statistical disclosure control and R.
sdcTable is free and open source and can be downloaded from CRAN (cran.r-project.org)
Everybody is encouraged to change, modify and to improve it.
Outline/Conclusions

- `sdcTable` is the first try to link statistical disclosure control and R.
- `sdcTable` is free and open source and can be downloaded from CRAN (cran.r-project.org)
- Everybody is encouraged to change, modify and to improve it.
- A lot of possibilities for future work exist.
Example: Protection tabular data using sdcTable

The End.

Thank you very much for your attention!