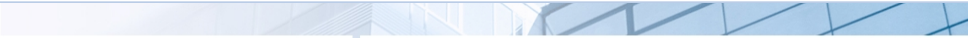
STATISTIK AUSTRIA
Die Informationsmanager



Johannes Gussenbauer,
Alexander Kowarik,
Bernhard Meindl
Statistik Austria
November, 2018

Implementation of the Cell-Key Method & Targeted Record Swapping

Introduction

- Cell-Key Method and Targeted Record Swapping implemented in R-Packages
- Available on `https://github.com/sdcTools`
  - `recordSwapping` (`https://github.com/sdcTools/recordSwapping`)
  - `cellKey` (`https://github.com/sdcTools/cellKey`)
- Implementations are prototype-ready

cellKey

- Two different ways to specify perturbation tables available:
  - ABS approach developed by Australian Bureau of Statistics
  - Approach developed by the Destatis
- `cellKey` depends on R-package `ptable` (https://github.com/sdcTools/ptable)

Main Features

- Methods `abs` and `destatis`
- Existing record-keys can be used or generated with `ck_generate_rkeys()`
- allows sampling weights
- perturbation of magnitude tables (for ABS-method only)
- main function `perturbTable()`
- useage of arbitrarily complex hierarchies like in `sdcTable`
- further functionality in `cellKey`
  - auxiliary methods (print, infoloss/utility, summary, ...) available
  - definition of binary sub-groups on the fly

Example

```
# load package
library(cellKey,verbose=FALSE)

## Loading required package: data.table

# load dummy data
dat <- ck_create_testdata()
dat <- dat[,c("sex","age","savings", "income","sampling_weight")]
dat[,cnt_highincome:=ifelse(income>=9000, 1, 0)]
```

→ create a perturbed table of counts of variables sex by age

Set parameters

- `pTable`: perturbation (lookup)-table for frequency table
- `sTable` and `mTable`: relevant input for perturbation of magnitude tables

```
pert_params <- ck_create_pert_params(
  bigN=17312941,
  smallN=12,
  pTable=ck_create_pTable(D=5, V=3, pTableSize=70, type="abs"),
  sTable=ck_generate_sTable(smallC=12),
  mTable=c(0.6,0.4,0.2))
```

Create input

```
inp <- ck_create_input(
  dat=dat,
  def_rkey=15*nrow(dat),
  pert_params=pert_params)
print(class(inp))

## [1] "pert_inputdat"
## attr(,"package")
## [1] "cellKey"
```

Specify Dimensions

```
# example for variable sex
dim.sex <- ck_create_node(total_lab="Total")
dim.sex <- ck_add_nodes(dim.sex, reference_node="Total",
  node_labs=c("male","female"))
print(dim.sex)

##    levelName
## 1 Total
## 2  Â¦--male
## 3  Â°--female
```

Specify Dimensions

```
dim.age <- ck_create_node(total_lab="Total")
dim.age <- ck_add_nodes(dim.age, reference_node="Total",
  node_labs=paste0("age_group",1:6))
print(dim.age)

##         levelName
## 1 Total
## 2  Â¦--age_group1
## 3  Â¦--age_group2
## 4  Â¦--age_group3
## 5  Â¦--age_group4
## 6  Â¦--age_group5
## 7  Â°--age_group6
```

Perturb Table

```
tab1 <- perturbTable(inp=inp, dimList=list(sex=dim.sex, age=dim.age),
  countVars="cnt_highincome",
  weightVar="sampling_weight", numVars=c("savings","income"))
print(tab1)

## The weighted 2-dimensional table consists of 21 cells. The results are
## The dimensions are given by the following variables
## o sex
## o age
##
## Type of pTable-used: 'abs'
## The following count-variables have been tabulated/perturbed:
## o Total
## o cnt_highincome
## The following numeric variables have been tabulated/perturbed:
## o savings
## o income
```

Perturbed Table

- return tables with `ck_freq_table()` or `ck_export_table()`

```
# count table containing
# original, perturbed and (un)weighted values
print(head(ck_export_table(tab1, vname="Total")))
```

```
##      sex        age vname  UWC     WC pUWC    pWC
## 1: Total      Total Total 4580 269665 4580 269665
## 2: Total age_group1 Total 1969 116010 1966 115833
## 3: Total age_group2 Total 1143  67615 1143  67615
## 4: Total age_group3 Total  864  50817  863  50758
## 5: Total age_group4 Total  423  24166  422  24109
## 6: Total age_group5 Total  168  10198  170  10319
```

- compute information loss measures with `ck_cnt_measures()`

```
ck_cnt_measures(tab1, vname="Total")
```

Perturbed Table

- perturbed table of continous (weighted) data

```
p_income <- ck_cont_table(tab1, vname="savings", meanBeforeSum=TRUE)
head(p_income, n=5)

##      sex       age UW_savings pUW_savings WS_savings pWS_savings
## 1: Total     Total    2273532   2268518.0  133862884   133567664
## 2: Total age_group1     982386    983131.6   57880447    57924379
## 3: Total age_group2     552336    549908.1   32673840    32530218
## 4: Total age_group3     437101    435993.0   25708520    25643351
## 5: Total age_group4     214661    213512.6   12263588    12197978
##    pWM_savings
## 1:    495.3096
## 2:    500.0680
## 3:    481.1095
## 4:    505.2081
## 5:    505.9512
```

Perturbed Table

- perturbed table for a specific group → by="cnt_highincome"

```
print(head(ck_export_table(tab1, vname="cnt_highincome")))

##       sex        age          vname UWC    WC pUWC   pWC
## 1: Total      Total cnt_highincome 445 26251  445 26251
## 2: Total age_group1 cnt_highincome 192 10755  191 10699
## 3: Total age_group2 cnt_highincome 123  7576  123  7576
## 4: Total age_group3 cnt_highincome  82  4826   84  4944
## 5: Total age_group4 cnt_highincome  34  2292   38  2562
## 6: Total age_group5 cnt_highincome  14   802   14   802
```

- More details and examples in the package vignette

```
vignette("introduction",package="cellKey")
```

Targeted Record Swapping

- Based on the SAS code on targeted record swapping from ONS
  - Some major difference between SAS and C++ implementation
- Implemented in C++11
  - C++ core functionality used by R-Package `recordSwapping` and Mu-Argus.
- single core-function `recordSwap()`

Main Function

```
recordSwap(data, # micro data
           similar, # variables considered when swapping
           hierarchy, # hierarchy levels
           risk, # risk variables
           th, # threshold for k-anonymity
           swaprate, # between 0 and 1
           seed # random seed
)
```

- `similar` only households with same household size are swapped
  - in prototype version procedure silently fails if no donor can be found
- count tables are generated using `risk` for each hierarchy
- Records which fullfil `counts` $\leq$ `th` are “high risk“ and must be swapped across respective hierarchy
- `swaprate` ~lower bound for swapped households

Example

```
library(recordSwapping)
# create some dummy data (~ 100k households)
dat <- recordSwapping:::create.dat(100000)
dat

##          nuts1 nuts2 nuts3 municipality    hid hsize ageGroup gender
##       1:     5    10     1            6      1     2        5      1
##       2:     5    10     1            6      1     2        6      2
##       3:     3     5     9           23      2     3        6      2
##       4:     3     5     9           23      2     3        6      2
##       5:     3     5     9           23      2     3        5      2
##      ---
## 350527:     3     5    14           15 100000     5        3      2
## 350528:     3     5    14           15 100000     5        1      1
## 350529:     3     5    14           15 100000     5        6      1
## 350530:     3     5    14           15 100000     5        6      2
## 350531:     3     5    14           15 100000     5        3      1
##          national htype hincome
##       1:        5     9       5
##       2:        5     9       5
```

Set Parameters

```r
colnames(dat)

##  [1] "nuts1"        "nuts2"        "nuts3"        "municipality"
##  [5] "hid"          "hsize"        "ageGroup"     "gender"
##  [9] "national"     "htype"        "hincome"
```

```r
# define paramters - in C++ indexing starts with 0 (!)
hierarchy <- 0:2 # nuts1 - nuts3
risk <- 5:7 # hsize - gender
hid <- 4 # column for hid
similar <- c(5) # hsize

# variables which are not column indices
swaprate <- .05 # swaprate of households
th <- 2 # counts <= th
```

Function Call

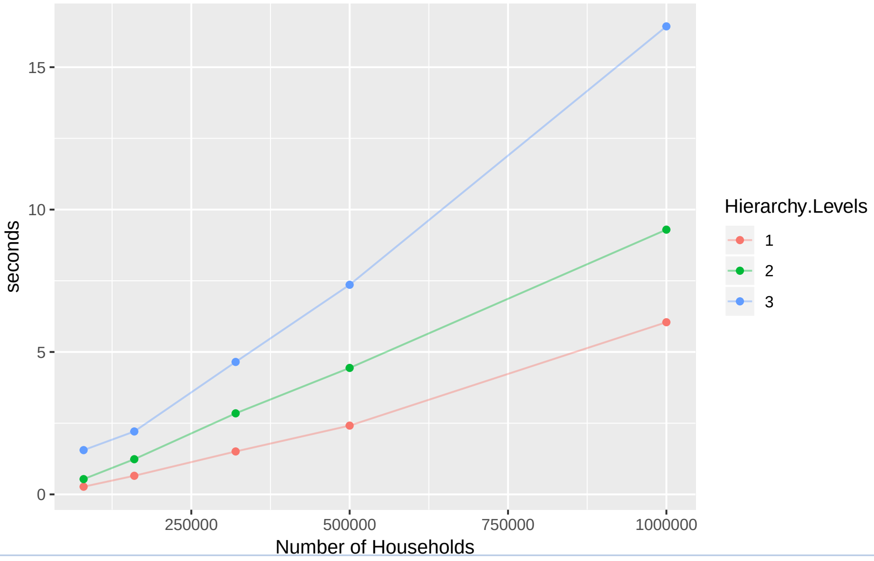
```
# call recodSwap()
dat_swapped <- recordSwap(dat,similar,hierarchy,risk,
                          hid,th,swaprate)
# returnes data with swapped records
dat_swapped

##         nuts1 nuts2 nuts3 municipality    hid hsize ageGroup gender
##      1:     5    10     1            6      1     2        5      1
##      2:     5    10     1            6      1     2        6      2
##      3:     3     5     9           23      2     3        6      2
##      4:     3     5     9           23      2     3        6      2
##      5:     3     5     9           23      2     3        5      2
##     ---
## 350527:     3     5    14           15 100000     5        3      2
## 350528:     3     5    14           15 100000     5        1      1
## 350529:     3     5    14           15 100000     5        6      1
## 350530:     3     5    14           15 100000     5        6      2
## 350531:     3     5    14           15 100000     5        3      1
##         national htype hincome
##      1:        5     9       5
```

Differences to SAS

- Arbitrary number of hierarchy levels and risk variables
- Risk is calculated using the combination of **all** risk variables
  - SAS-Code uses each risk variable seperately
- Sampling probability is defined by $\frac{1}{counts}$
- Number of swaps households are distributed proportional to size
- "high risk" households are mandatorily swapped
  - set `th <- 0` to disable this
- More details in the package vignette

```
vignette("recordSwapping")
```

Benchmark



seconds
15
10
5
0
250000
500000
750000
1000000
Number of Households
Hierarchy.Levels
1
2
3

Outlook for Record Swapping

- Supply risk from external source
- Multiple similarity profiles
- Return information if donor cannot be found
- Add utility measure based on the spatial correlation
- Supply either risk threshold or swaprate