# Implementation of the Cell-Key Method & Targeted Record Swapping

**Johannes Gussenbauer**,
Alexander Kowarik,
Bernhard Meindl
Statistik Austria

November, 2018

# 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

```r
# 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 273815 4580 273815
## 2: Total age_group1 Total 1969 117585 1970 117645
## 3: Total age_group2 Total 1143  69057 1148  69359
## 4: Total age_group3 Total  864  51160  864  51160
## 5: Total age_group4 Total  423  24514  422  24456
## 6: Total age_group5 Total  168  10640  168  10640
```

- 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   2274882.8  135922962   136003717
## 2: Total age_group1     982386    982463.7   58666256    58670893
## 3: Total age_group2     552336    551587.9   33370662    33325464
## 4: Total age_group3     437101    437888.4   25882045    25928668
## 5: Total age_group4     214661    214251.6   12440189    12416463
##    pWM_savings
## 1:    496.6993
## 2:    498.7113
## 3:    480.4779
## 4:    506.8152
## 5:    507.7062
```

# 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 26723  446 26783
## 2: Total age_group1 cnt_highincome 192 11635  193 11696
## 3: Total age_group2 cnt_highincome 123  7319  124  7379
## 4: Total age_group3 cnt_highincome  82  4986   85  5168
## 5: Total age_group4 cnt_highincome  34  1846   35  1900
## 6: Total age_group5 cnt_highincome  14   937   14   937
```

- 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

```r
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` $\sim$lower bound for swapped households

# Example

```
library(recordSwapping)

## Error in library(recordSwapping): there is no package called
'recordSwapping'

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

## Error in loadNamespace(name): there is no package called
'recordSwapping'

dat

##           sex        age savings income sampling_weight cnt_highincome
##    1:   male age_group3      12   5780              49              0
##    2: female age_group3      28   2530              49              0
##    3:   male age_group1     550   6920              34              0
##    4:   male age_group1     870   7960              70              0
##    5:   male age_group4      20   9030              40              1
##   ---
## 4576: female age_group3     278   7900              83              0
```

# Set Parameters

```
colnames(dat)

## [1] "sex"             "age"             "savings"         "income"
## [5] "sampling_weight" "cnt_highincome"
```

```
# 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)

## Error in recordSwap(dat, similar, hierarchy, risk, hid, th,
swaprate): could not find function "recordSwap"

# returnes data with swapped records
dat_swapped

## Error in eval(expr, envir, enclos): object 'dat_swapped' not found
```

# 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

```
## Error in find.package(package, lib.loc, verbose = verbose): there is
no package called 'recordSwapping'
## Error in rbindlist(mb_all): object 'mb_all' not found
## Error in eval(expr, envir, enclos): object 'mb_all' not found
## Error in eval(expr, envir, enclos): object 'mb_all' not found
## Error in ggplot(mb_all, aes(npop, value)): object 'mb_all' not found
## Error in plot(p1): object 'p1' not found
```

# Outlook for Record Swapping

- Supply risk from external source
- Multiple similarity profiles
- Return information if donor cannot be found
- Add function to calculate information loss
- Supply either risk threshold or swaprate