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`
  - `cellKey`
- Implementations are prototype-ready

- Two different approaches implemented
  - *Methodology for the Automatic Confidentialisation of Statistical Outputs from Remote Servers at the Australian Bureau of Statistics* (Thompson, Broadfoot, Elazar)
  - Approach developed by the Federal Statistical Office of Germany (Destatis)
- `cellKey` depends on R-package `ptable`

Main Features

- Methods `abs` and `destatis`
- `ck_generate_rkeys()` for generating record keys
  - perturbation for magnitude tables only for `abs`
- main function `pertubTable()`
- allows sampling weights
- define arbitrarily complex hierarchies like in `sdcTable`
  - improved functionality in `cellKey`
- various auxiliary methods implemented

Example

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

## Loading required package: data.table

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

$\rightarrow$ create a perturbed table of counts of variables sex by age

Set parameters

```
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 <- data.table(levels=c("@","@@","@@"),
                      codes=c("Total", "male", "female"))
print(dim.sex)

##    levels  codes
## 1:      @  Total
## 2:     @@   male
## 3:     @@ female
```

```
# or alternatively
dim.sex2 <- ck_create_node(total_lab="Total")
dim.sex2 <- ck_add_nodes(dim.sex2, reference_node="Total",
                         node_labs=c("male","female"))
print(dim.sex2)

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

Specify Dimensions

```r
# example for variable age
dim.age <- data.table(levels=c("@",rep("@@", 6)),
                      codes=c("Total", paste0("age_group",1:6)))

# or alternatively
dim.age2 <- ck_create_node(total_lab="Total")
dim.age2 <- ck_add_nodes(dim.age2, reference_node="Total",
                         node_labs=paste0("age_group",1:6))
print(dim.age2)

##         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
```

Compine Hierachies

```
dimList <- list(sex=dim.sex, age=dim.age2)
print(dimList)

## $sex
##    levels  codes
## 1:      @  Total
## 2:     @@   male
## 3:     @@ female
##
## $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=dimList,
                     countVars=c("cnt_males", "cnt_highincome"),
  weightVar="sampling_weight", numVars=NULL)
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_males
## o cnt_highincome
## No numeric variables have been tabulated/perturbed in this table
```

Perturbed Table

- returns tables with `ck_freq_table()`

```
# count table containing
# original, perturbed and (un)weighted values
ck_freq_table(tab1, vname="cnt_males")
```

- compute information loss measures with `ck_cnt_measures()`

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

Perturbed Table

- perturbed table of continous data

```
tab2 <- perturbTable(inp=inp,dimList=dimList,weightVar="sampling_weight",
  countVars=c("cnt_males", "cnt_highincome"),
  numVars=c("savings","income"))

p_income <- ck_cont_table(tab2, vname="income", meanBeforeSum=TRUE)
head(p_income)
```

```
##      sex         age UW_income pUW_income  WS_income pWS_income pWM_inco
## 1: Total       Total  22952978 22966209.6 1364098489 1364884844   5013.3
## 2: Total age_group1   9810547  9802020.2  590600908  590087587   4968.0
## 3: Total age_group2   5692119  5685073.1  341208421  340786063   4978.1
## 4: Total age_group3   4406946  4431279.5  254404219  255808944   5134.7
## 5: Total age_group4   2133543  2132167.1  125571363  125490386   5016.8
## 6: Total age_group5    848151   859507.7   47945774   48587766   5085.5
```

Perturbed Table

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

```
tab3 <- perturbTable(inp=inp, dimList=dimList,
  weightVar="sampling_weight",
  numVars=c("savings"), by="cnt_highincome")
```

- 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 nuts4    hid hsize ageGroup gender national
##      1:     4     4     5    21      1     1         3      2        1
##      2:     5     9     1     1      2     2         3      1        1
##      3:     5     9     1     1      2     2         4      1        3
##      4:     3     9    13    12      3     5         3      2        1
##      5:     3     9    13    12      3     5         3      2        3
##     ---
## 349846:     2     3     3    24  99999     2         3      1        2
## 349847:     2     4     3    25 100000     4         1      2        2
## 349848:     2     4     3    25 100000     4         6      2        1
## 349849:     2     4     3    25 100000     4         2      1        1
## 349850:     2     4     3    25 100000     4         1      2        1
##          htype hincome
##      1:     4        1
##      2:     6        3
```

Set Parameters

```
colnames(dat)

##  [1] "nuts1"    "nuts2"    "nuts3"    "nuts4"    "hid"      "hsize"
##  [7] "ageGroup" "gender"   "national" "htype"    "hincome"
```

```
# 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
```

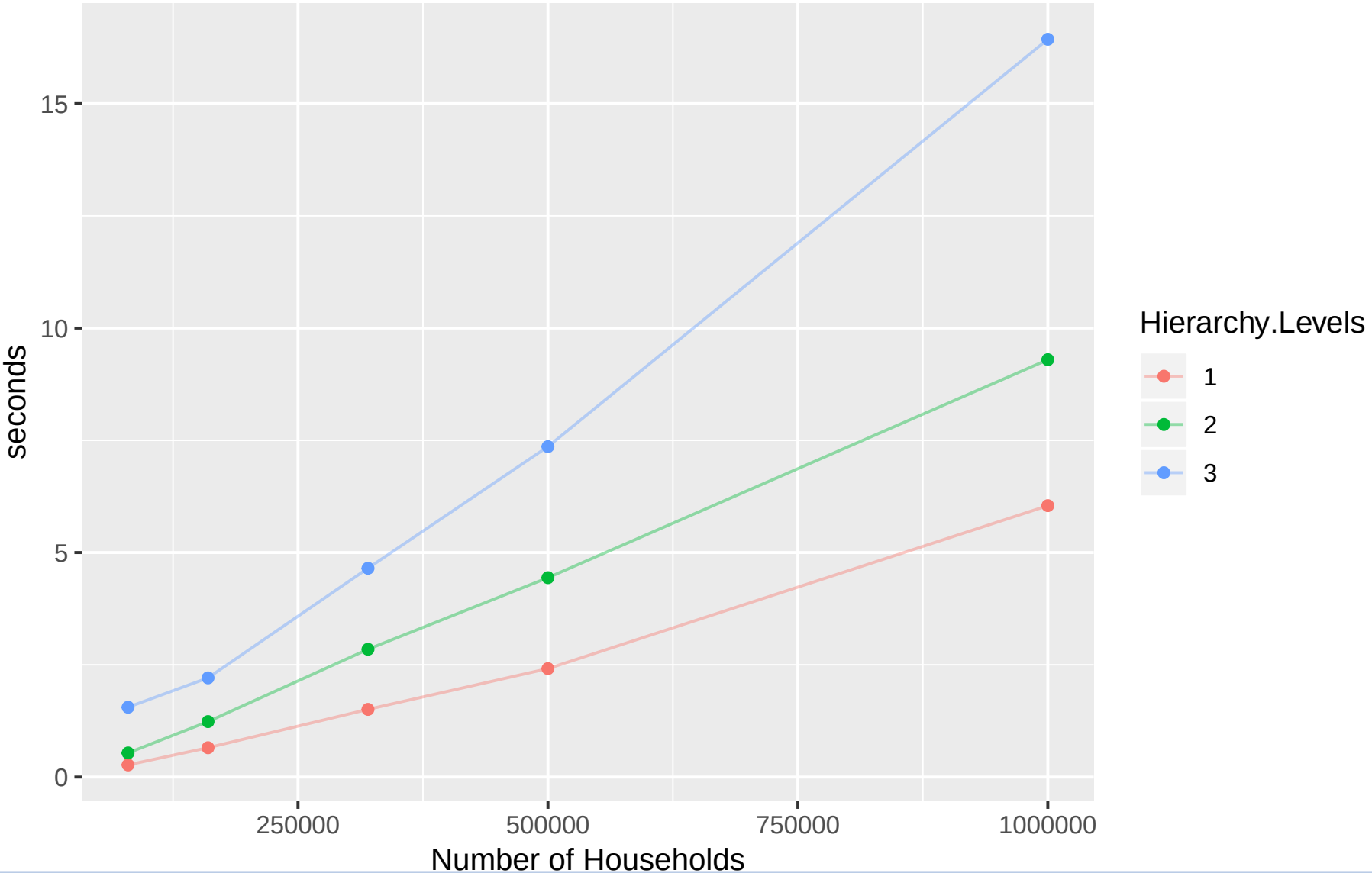
```
##          nuts1 nuts2 nuts3 nuts4    hid hsize ageGroup gender national
##      1:     4     7     5    21      1     1         3      2        1
##      2:     2     2    10     1      2     2         3      1        1
##      3:     2     2    10     1      2     2         4      1        3
##      4:     3     9    13    12      3     5         3      2        1
##      5:     3     9    13    12      3     5         3      2        3
##     ---
## 349846:     4     2     1    24  99999     2         3      1        2
## 349847:     2     4     3    25 100000     4         1      2        2
## 349848:     2     4     3    25 100000     4         6      2        1
## 349849:     2     4     3    25 100000     4         2      1        1
## 349850:     2     4     3    25 100000     4         1      2        1
##          htype hincome
##      1:     4        1
```

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


15
10
5
0
seconds
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 function to calculate information loss
- Supply either risk threshold or swaprate