

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

Implementation of the Cell-Key Method & Targeted Record Swapping

- ▶ 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

- ▶ 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>)

- ▶ 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

```
# 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

- ▶ 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))
```

```
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"
```

```
# 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  Ã|--male
## 3  Ã°--female
```



```
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
```

```
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
```

- ▶ 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 274020 4582 274140  
## 2: Total age_group1 Total 1969 118371 1971 118491  
## 3: Total age_group2 Total 1143  67585 1142  67526  
## 4: Total age_group3 Total   864  51812  865  51872  
## 5: Total age_group4 Total   423  24973  422  24914  
## 6: Total age_group5 Total   168  10574  167  10511
```

- ▶ compute information loss measures with `ck_cnt_measures()`

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

- ▶ 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   2273887.4  136024725  136045987
## 2: Total age_group1    982386    983801.7   59058412   59143520
## 3: Total age_group2    552336    551111.0   32659343   32586910
## 4: Total age_group3    437101    438825.5   26211895   26315311
## 5: Total age_group4    214661    215164.0   12673119   12702816
##      pWM_savings
## 1:      496.2646
## 2:      499.1393
## 3:      482.5831
## 4:      507.3125
## 5:      509.8666
```

- ▶ 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	26694	446	26754
## 2:	Total	age_group1	cnt_highincome	192	11652	194	11773
## 3:	Total	age_group2	cnt_highincome	123	7486	123	7486
## 4:	Total	age_group3	cnt_highincome	82	4808	84	4925
## 5:	Total	age_group4	cnt_highincome	34	1918	35	1974
## 6:	Total	age_group5	cnt_highincome	14	830	14	830

- ▶ More details and examples in the package vignette

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

- ▶ 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()

```
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 fulfill counts \leq th are “high risk” and must be swapped across respective hierarchy
- ▶ swaprate ~lower bound for swapped households

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

```
##          nuts1 nuts2 nuts3 nuts4      hid hsize ageGroup gender national
##      1:         5     2     5    25         1     1         2         2         3
##      2:         4     6    15     7         2     1         2         2         5
##      3:         1     4     2    18         3     5         1         1         5
##      4:         1     4     2    18         3     5         1         1         3
##      5:         1     4     2    18         3     5         3         1         2
##      ---
## 349468:         1     7     4     7 99999         2         5         1         5
## 349469:         2    10    10    17 100000         4         5         1         3
## 349470:         2    10    10    17 100000         4         5         2         5
## 349471:         2    10    10    17 100000         4         4         2         4
## 349472:         2    10    10    17 100000         4         3         1         3
##          htype hincome
##      1:         7         6
```



```
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
```

```
swaprte <- .05 # swaprte of households
```

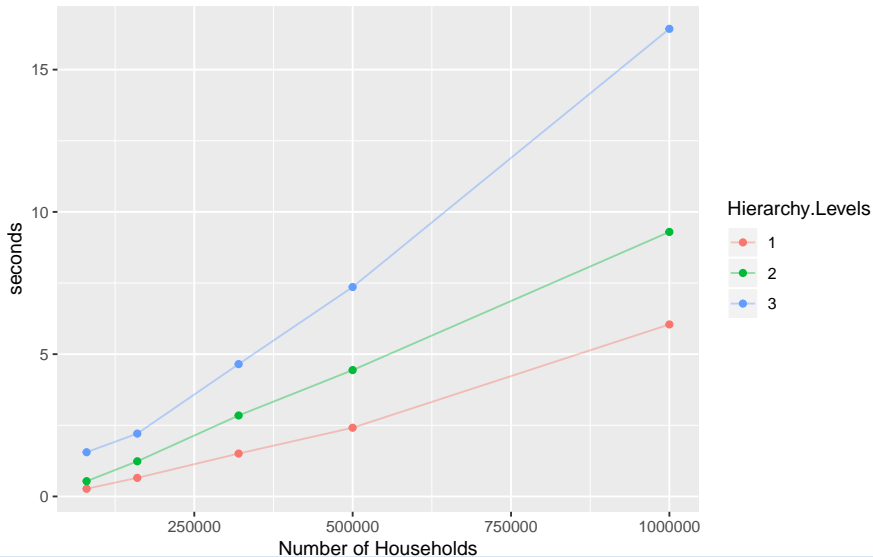
```
th <- 2 # counts <= th
```

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

```
##          nuts1 nuts2 nuts3 nuts4      hid hsize ageGroup gender national
##      1:      5      8      7     25        1      1          2        2          3
##      2:      4      6     15      7        2      1          2        2          5
##      3:      1      4      2     18        3      5          1        1          5
##      4:      1      4      2     18        3      5          1        1          3
##      5:      1      4      2     18        3      5          3        1          2
##      ---
## 349468:      3      4      7      7 99999          2          5          1          5
## 349469:      2     10     10     17 100000         4          5          1          3
## 349470:      2     10     10     17 100000         4          5          2          5
## 349471:      2     10     10     17 100000         4          4          2          4
## 349472:      2     10     10     17 100000         4          3          1          3
##          htype hincome
```

- ▶ Arbitrary number of hierarchy levels and risk variables
- ▶ Risk is calculated using the combination of **all** risk variables
 - ▶ SAS-Code uses each risk variable separately
- ▶ 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")
```



- ▶ 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