Bernhard Meindl
Tobias Enderle
Statistics Austria / Destatis
The Hague
October 2019

Consistently perturbing tables
Introducing R package cellKey
What (in a nutshell)

- Presenting R package cellKey ([github.com/sdcTools/cellKey](https://github.com/sdcTools/cellKey))
- Developed within project “Open Source tools for perturbative confidentiality methods”
  - Thx to all colleagues from Germany, Netherlands, France, Finland, Slovenia, Finland and Hungary
- The tool implements the (enhanced) cell-key method
Features:

- Starting point: microdata
  - importat: assignment of record keys (rkeys)
  - allows to easily define (complex) tables
  - allows to perturb count- and magnitude tables using cell keys (ckeys) derived from rkeys
- (Sampling)weights are supported
- Allows for different perturbation parameters by variable
The method: Starting from micro data

- **Idea:** describing the general idea of the method (easy)
- Post-tabular method when other methods (e.g., suppression) are not viable
- starting from a micro data set `md` (nrows: 4580)

```r
print(head(md))
```

```r
## sex age high_inc w savings
## 1: male ag3 0 93 12
## 2: female ag3 0 97 28
## 3: male ag1 0 42 550
## 4: male ag1 0 89 870
## 5: male ag4 1 23 20
## 6: female ag3 0 74 102
```
The method: Defining record keys

- adding record keys (assumed to be $\in (0; 1]$)
- auxiliary function `ck_generate_rkeys()` is available

```r
md$rkey <- cellKey::ck_generate_rkeys(md); print(head(md))
```

```
## sex age high_inc w savings rkey
## 1: male ag3 0 93 12 0.2954497
## 2: female ag3 0 97 28 0.4023517
## 3: male ag1 0 42 550 0.3217148
## 4: male ag1 0 89 870 0.2718497
## 5: male ag4 1 23 20 0.3701754
## 6: female ag3 0 74 102 0.6838545
```
The method: Identify contributors to a cell

Computing a cell key:
- summation of rkeys for all units contributing to the cell

```
##  sex  age high_inc  w savings   rkey
## 1: male ag6   0  45    771  0.8911682
## 2: male ag6   0  64    767  0.5697663
## 3: male ag6   0  57    821  0.4637136
## 4: male ag6   0  54    41   0.7340508
## 5: male ag6   0  30    281  0.7369458
## 6: male ag6   0  84    558  0.9178036
## 7: male ag6   0  44    371  0.9395674
```
The method: Compute the cell key

- The cell key is computed as
  - sum of the individual record keys
  - taking the remainder of a division by 1 of this value
- → the cell keys are also $\in [0, 1)$

---

**Important**: as the same ckey is computed for each cell with an identical set of contributors, we can easily achieve consistency

**Idea**:  
- we use the cell keys to find a perturbation values  
- we achieve consistency and loose additivity
The method: Perturbing the cell (1)

- **Trivial:** using the `ckey` as seed when sampling a perturbation value (i.e. noise)

```markdown
## 1 3
```

- This noise will always be 3 for any table cell to which the same set of units contributes.
we can then use this noise to perturb the cell count

```
## [1] 378
```

and receive the perturbed and, hence, protected cell count

```
## [1] 381
```

**However:** How or where can we find the noise of 3 that will be added to the cell count 378 using the ckey 0.2530157?

For the lookup of the noise we need a perturbation table . . .
The ptable package (1)

- The CKM (and thus also cellKey) depends on perturbation tables.
- These so-called ptables encompass noise values and the corresponding probabilities.
- There are ptables for both: frequency and magnitude tables.
- The ptables need to be designed such that:
  - no bias is introduced
  - the variance of the noise is fixed
  - in case of frequency tables:
    - no negative counts will appear
    - zero counts remain zeroes
ptables can be computed with R package ptable (github.com/sdcTools/ptable)

The ptable results from the solution of non-linear equation systems that are defined by constraints and boundary conditions relying on a maximum entropy optimization

**Installation** and **Loading** as easy as:

```r
remotes::install_github(
  repo = "sdcTools/ptable",
  dependencies = TRUE)

library(ptable)
```
The package allows for a couple of arguments - used for the constraints of the optimization instance - when designing the ptable:

- $D$ maximum noise
- $V$ variance
- $p_{stay}$ probability that a frequency count remains unchanged/unperturbed
- $j$ small counts can be blocked (i.e. they will not appear in the protected table)
- ... and many more ...

Vignette gives an introduction with examples: call `pt_vignette()`
Example: Design the ptable for a frequency table with maximum noise $D=2$, a fixed variance $V=1.05$, counts of 1 shall not appear in the protected tables and the probability that a frequency count is set to 50%.

```r
# Define the parameters
ptab_params <- pt_create_pParams(D=2,
                                V=1.05,
                                js = 1,
                                pstay=0.5,
                                mono = c(T,T,F,T),
                                table = "cnts")

# Compute the ptable
ptab <- pt_create_pTable(params = ptab_params)
```
Example (continued)

```r
# Evaluate the result
ptab@empResults
```

```
##         i   p_mean   p_var   p_sum   p_stay  iter
## 1:      0.00 0.0000 1.0000   1.000   0.000    0
## 2:      1.05 1.0500 1.0500   1.050   0.000    1
## 3:      1.05 1.0500 1.0500   1.050   0.555    1
## 4:      1.05 1.0500 1.0500   1.050   0.288    6
## 5:      1.05 1.0500 1.0500   1.050   0.500    1
```
Example (continued)

```r
# Look at the ptable
head(ptab@pTable)
```

```
## i j p v p_int_lb p_int_ub type
## 1: 0 0 1.00000000 0 0.0000000 1.0000000 all
## 2: 1 0 0.50833333 -1 0.0000000 0.5083333 all
## 3: 1 2 0.47500000 1 0.5083333 0.9833333 all
## 4: 1 3 0.01666667 2 0.9833333 1.0000000 all
## 5: 2 0 0.16155827 -2 0.0000000 0.1615583 all
## 6: 2 2 0.55565037 0 0.1615583 0.7172086 all
```
Example (continued)

```r
# Plot the ptable: perturbation panel
plot(ptab, type="p")
```

## Perturbation Panel

### Perturbation Panel

- `p` (probability)
- `i` (original frequency)
- `v` (perturbation value):
  - 0 (no perturbation)
  - ±1
  - ±2

---

Obtaining perturbation tables

slide 16 | October 2019
Getting started with the cellKey package

- **Installation** as easy as:
- (online) **Dokumentation**: sdctools.github.io/cellKey
  - includes (executable) examples
  - comes with a package vignette showing a complete example
Defining tables is often difficult due to (complex hierarchies)
cellKey uses functionality from sdcHierarchies
Computation from (character) vector:

```
##  level  name
##   1   @ total
##   2   @@ a
##   3  @@@ a1
##   4  @@@ a2
##   5   @@ b
##   6  @@@ b1
##   7  @@@ b2
##   8   @@ c
```
direct setup is possible too

```r
## level name
## 1 @ total
## 2 @@ a
## 3 @@@ a1
## 4 @@@ a2
## 5 @@ b
## 6 @@@ b1
## 7 @@@ b2
## 8 @@ c
```
In order to define a table instance, we need:

- micro data
- hierarchies
- optional count- and continuous variables for later perturbation

This makes use of functionality from `sdcTable`

We can use function `ck_setup()`
we define the hierarchies for this example:
and setup the table-instance
objects from `ck_setup` have their data and methods bundled together

methods can be called with the following syntax

for example (no perturbed results yet):

```
## sex age vname uwc wc puwc pwc
## 1: total total total 4580 272676 NA NA
## 2: total ag1 total 1969 118424 NA NA
## 3: total ag2 total 1143 67188 NA NA
## 4: total ag3 total 864 50112 NA NA
## 5: total ag4 total 423 25946 NA NA
## 6: total ag5 total 168 10239 NA NA
```
In order to perturb variables, we need to attach parameters to variables

Auxiliary functions:
- for count variables: ck_params_cnts()
- for continuous variables: ck_params_nums()

Example: Parameters for a count variable
Attaching parameters to a variable is done with `params_cnts_set()` method (by default for all variables)

```r
# --> setting perturbation parameters for variable 'total'
# --> setting perturbation parameters for variable 'high_inc'
```

for specific variables

```r
# --> replacing perturbation parameters for variable 'high_inc'
```
Attaching parameters for numerical variables (magnitude) tables

- ck_params_nums() is used to create a suitable input
- params_nums_set() is used to attach the parameters to variables

```r
# --> setting perturbation parameters for variable 'savings'
```
- actually perturbing variables can be achieved by using the `perturb` method
- it is the same for count vars

```r
## Count variable 'total' was perturbed.
```

```r
## Count variable 'high_inc' was perturbed.
```

- and numerical variables

```r
## Numeric variable 'savings' was perturbed.
```
we can use the `freqtab()` and `numtab()` methods to extract results

```r
## sex age vname uwc wc puwc pwc
## 1: total total high_inc 445 26467 442 26288.5708
## 2: total ag1 high_inc 192 11717 193 11778.0260
## 3: total ag2 high_inc 123 7236 123 7236.0000
## 4: total ag3 high_inc 82 4417 80 4309.2683
## 5: total ag4 high_inc 34 2053 34 2053.0000
## 6: total ag5 high_inc 14 1044 13 969.4286

## sex age vname uws ws pws
## 1: total total savings 2273532 136022592 136026830
## 2: total ag1 savings 982386 59293043 59291584
## 3: total ag2 savings 552336 32850275 32848975
## 4: total ag3 savings 437101 25219456 25219438
## 5: total ag4 savings 214661 13274444 13273339
## 6: total ag5 savings 80451 4975963 4975637
```
CKM was extended quite a lot for magnitude tables

- different "starting values" for perturbation possible
  - cellmean, range, cell value, top contributors

- possibility to use different ptables for
  - cells with even/odd number of contributors
  - for very small cells

- allow for extra perturbation for sensitive cells
  - some methods (via supp_{method}()) are implemented
distance/utility measures for count-variables (measures_cnts())

perturbation parameters can be saved for reproducability

- ck_write_yaml() to write parameters
- ck_read_yaml() to import parameters

saving tables: freqtab() and numtab() have an optional parameter path
cellKey: future development

- incorporating **feedback** from users
- add utility statistics and measures for continuous variables
- achieve full consistency with the implementation from $\tau$-argus
cellKey allows to consistently perturb statistical tables
it relies on ptables created from the ptable package

Important Links

- Documentation: sdctools.github.io/cellKey/
- Questions: github.com/sdcTools/userSupport
- Bug-Reports: github.com/sdcTools/userSupport/issues
Consistently perturbing tables
Introducing R package cellKey