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

- Presenting R package cellKey (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
  - originally proposed by ABS
  - post-tabular method to protect statistical tables
  - enhanced/modified within this project
Features:

- Starting point: microdata
  - important: assignment of record keys (rkeys)
  - allows to easily define (complex) tables
  - allows to perturb count- and magnitude tables using cell keys (ckkeys) derived from rkeys
- (Sampling)weights are supported
- Allows for different perturbation parameters by variable
- many auxiliary functions (e.g. key generation) available
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`

```r
# get microdata from github
url <- "https://git.io/JezKG"
md <- data.table::fread(url, showProgress = FALSE)
print(head(md))
```

<table>
<thead>
<tr>
<th></th>
<th>sex</th>
<th>age</th>
<th>high_inc</th>
<th>w</th>
<th>savings</th>
<th>rkey</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>male</td>
<td>ag3</td>
<td>0</td>
<td>70</td>
<td>12</td>
<td>0.8027299</td>
</tr>
<tr>
<td>2</td>
<td>female</td>
<td>ag3</td>
<td>0</td>
<td>99</td>
<td>28</td>
<td>0.6282568</td>
</tr>
<tr>
<td>3</td>
<td>male</td>
<td>ag1</td>
<td>0</td>
<td>58</td>
<td>550</td>
<td>0.3713780</td>
</tr>
<tr>
<td>4</td>
<td>male</td>
<td>ag1</td>
<td>0</td>
<td>23</td>
<td>870</td>
<td>0.2507976</td>
</tr>
<tr>
<td>5</td>
<td>male</td>
<td>ag4</td>
<td>1</td>
<td>38</td>
<td>20</td>
<td>0.9165093</td>
</tr>
<tr>
<td>6</td>
<td>female</td>
<td>ag3</td>
<td>0</td>
<td>93</td>
<td>102</td>
<td>0.8307836</td>
</tr>
</tbody>
</table>
The method: Deriving a cell key (1)

- Identify contributors for a specific cell:

```r
contributors <- md[sex == "male" & age == "ag6"]
contributors
```

```
## sex age high_inc w savings rkey
## 1: male ag6 0 29 771 0.11134575
## 2: male ag6 0 90 767 0.75515496
## 3: male ag6 0 38 821 0.56953559
## 4: male ag6 0 26 41 0.45790072
## 5: male ag6 0 59 281 0.16707917
## 6: male ag6 0 39 558 0.05877474
## 7: male ag6 0 81 371 0.06648822
```

- we have `nrow(contributors)` to this cell
The method: Deriving a cell key (2)

- **ckey**: Sum of all contributing record keys and taking the fractal part
  - → the cell keys are $\sim \in [0, 1)$

```r
ckey <- sum(contributors$rkey) %% 1; ckey
```

```r
## [1] 0.1862791
```

- **Consistency**: the same set of units returns the same ckey
- **Perturbation**: find a perturbation value based on the ckey
The method: Perturbing the cell

- **Trivial**: using the ckey as seed when sampling a perturbation value

```r
pval <- function(ckey) { set.seed(ckey); sample(-10:10, 1) }
```

- **Perturbation**: we use the noise value to perturb the cell count

```r
result <- sum(contributors$w) + pval(ckey); result
```

```r
## [1] 365
```

- **Task**: Finding perturbation values → make use of a perturbation table
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
- **pstay** probability that a frequency count remains unchanged/unperturbed
- **js** 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
ptab <- ptable::create_cnt_ptable(
  D = 2,
  V = 1.05,
  js = 1,
  pstay = 0.5,
  mono = c(T, T, F, T))
```
The ptable package (5)

Example (continued)

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

```
## i  p_mean  p_var  p_sum  p_stay  iter
## 1: 0  0.00   0.00   1.00    1.0000   0
## 2: 1  1.05   1.05   1.05    0.0000   1
## 3: 2  1.05   1.05   1.05    0.5557   1
## 4: 3  1.05   1.05   1.05    0.2881   6
## 5: 4  1.05   1.05   1.05    0.5000   1
```
Example (continued)

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

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>p</th>
<th>v</th>
<th>p_int_lb</th>
<th>p_int_ub</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1.00000000</td>
<td>0</td>
<td>0.0000000</td>
<td>1.0000000</td>
<td>all</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.50833333</td>
<td>-1</td>
<td>0.0000000</td>
<td>0.5083333</td>
<td>all</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.47500000</td>
<td>1</td>
<td>0.5083333</td>
<td>0.9833333</td>
<td>all</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.01666667</td>
<td>2</td>
<td>0.9833333</td>
<td>1.0000000</td>
<td>all</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.16155827</td>
<td>-2</td>
<td>0.0000000</td>
<td>0.1615583</td>
<td>all</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0.55565037</td>
<td>0</td>
<td>0.1615583</td>
<td>0.7172086</td>
<td>all</td>
</tr>
</tbody>
</table>
Example (continued)

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

## Perturbation Panel

```
0
1
2
3
4+
0.00 0.25 0.50 0.75 1.00
p (probability)
i (original frequency)
v (perturbation value):
−2 −1 0 (no perturbation) +1 +2
```

Obtaining perturbation tables
Getting started with the cellKey package

Installation
- directly from the github.com/sdcTools/cellKey (not yet on CRAN)

(online) Dokumentation: sdctools.github.io/cellKey
- includes a package vignette

defining hierarchies
- cellKey uses functionality from sdcHierarchies
- hierarchies can be created
  - indirectly (from a character vector)
  - directly (by adding nodes to a tree)
**Prerequisites**: micro data, hierarchies, optionally count and continuous variables, weights and record keys

**Dimensions**

```r
dims <- list(
    sex = hier_create("total", nodes = c("male", "female")),
    age = hier_create("total", nodes = paste0("ag", 1:6)))
```

**Table instance**

```r
tab <- ck_setup(x = md, rkey = "rkey", dims = dims,
                w = "w", countvars = c("high_inc"), numvars = "savings")
```
objects created with `ck_setup()` have their data and methods bundled together

methods can be called with the following syntax

```r
obj$method(...)"
```

for example (no perturbed results yet):

```r
head(tab$freqtab(v = "total"))
```

```
## sex age vname uwc    wc puwc   pwc
## 1: total total total 4580 274917 NA   NA
## 2: total ag1 total 1969 119138 NA   NA
## 3: total ag2 total 1143  68337 NA   NA
## 4: total ag3 total  864  51841 NA   NA
## 5: total ag4 total  423  24759 NA   NA
## 6: total ag5 total  168  10105 NA   NA
```
Attaching parameters to variables:

- for count variables: `ck_params_cnts()`
- for continuous variables: `ck_params_nums()`

Example: Parameters for a count variable

```r
library(ptable)
cnt_params1 <- ck_params_cnts(ptab = create_cnt_ptable(D = 8, V = 3, js = 2))
cnt_params2 <- ck_params_cnts(ptab = create_cnt_ptable(D = 10, V = 10, js = 5))
```
cellKey: attaching perturbation parameters (2)

- **Attaching** parameters to variable(s)
  - `params_cnts_set()` / `params_nums_set()`

```r
tab$params_cnts_set(cnt_params1) # default: all vars
```

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

- for specific variables (resetting previous parameters)

```r
tab$params_cnts_set(v = "high_inc", val = cnt_params2)
```

```r
## --> replacing perturbation parameters for variable 'high_inc'
```
cellKey: Perturbing variables and returning results

- **Perturbation**: use the `perturb` method (for both count- and continuous variables)

```r
tab$perturb(c("total", "high_inc"))
```

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

- **Results**: use the `freqtab()` and `numtab()` methods

```r
## sex age vname uwc wc puwc pwc
## 1: total total high_inc 445 26535 444 26475.3708
## 2: total ag1 high_inc 192 11881 190 11757.2396
## 3: total ag2 high_inc 123 6833 122 6777.4472
## 4: total ag3 high_inc 82 4980 78 4737.0732
## 5: total ag4 high_inc 34 1967 33 1909.1471
## 6: total ag5 high_inc 14 874 12 749.1429
```
CKM was extended quite a lot for magnitude tables

- different “starting values” for perturbation possible
- apply different ptables for cells with
  - an even/odd number of contributors
  - very small values
- allow for extra perturbation for sensitive cells
  - `supp_{method}()` namespace for identification

- distance/utlity measures for count-variables (`measures_cnts()`)
- perturbation parameters can be saved for reproducability
  (`ck_write_yaml()` and `ck_read_yaml()`)

www.statistik.at Perturbing tables
Future development:

- incorporating feedback from users
- add utility statistics and measures for continuous variables
- achieve full consistency with the implementation from $\tau$-argus

Summary

- cellKey allows to consistently perturb statistical tables
- it makes use of 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