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
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 (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))
```

```
## sex age high_inc  w savings
## 1: male ag3   0   70    12
## 2: female ag3 0   99    28
## 3: male ag1   0   58   550
## 4: male ag1   0  23   870
## 5: male ag4   1  38    20
## 6: female ag3 0  93   102
```
The method: Defining record keys

- adding record keys (assumed to be $\in (0; 1]$)
- auxiliary function \texttt{ck}\_generate\_rkeys() is available

```r
md$rkey <- cellKey::ck\_generate\_rkeys(md); 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: Identify contributors to a cell

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

<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>##</td>
<td>1:</td>
<td>male</td>
<td>ag6</td>
<td>0</td>
<td>29</td>
<td>771 0.11134575</td>
</tr>
<tr>
<td>##</td>
<td>2:</td>
<td>male</td>
<td>ag6</td>
<td>0</td>
<td>90</td>
<td>767 0.75515496</td>
</tr>
<tr>
<td>##</td>
<td>3:</td>
<td>male</td>
<td>ag6</td>
<td>0</td>
<td>38</td>
<td>821 0.56953559</td>
</tr>
<tr>
<td>##</td>
<td>4:</td>
<td>male</td>
<td>ag6</td>
<td>0</td>
<td>26</td>
<td>41 0.45790072</td>
</tr>
<tr>
<td>##</td>
<td>5:</td>
<td>male</td>
<td>ag6</td>
<td>0</td>
<td>59</td>
<td>281 0.16707917</td>
</tr>
<tr>
<td>##</td>
<td>6:</td>
<td>male</td>
<td>ag6</td>
<td>0</td>
<td>39</td>
<td>558 0.05877474</td>
</tr>
<tr>
<td>##</td>
<td>7:</td>
<td>male</td>
<td>ag6</td>
<td>0</td>
<td>81</td>
<td>371 0.06648822</td>
</tr>
</tbody>
</table>
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
- $\rightarrow$ the cell keys are also $\in [0, 1)$

## [1] 0.1862791

- **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)

  ```
  ## [1] 3
  ```

- this noise will always be 3 for any table cell to which the same set of units contributes.
The method: Perturbing the cell (2)

- we can then use this noise to perturb the cell count
  
  ```
  ## [1] 362
  ```

- and receive the perturbed and, hence, protected cell count

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

- **However:** How or where can we find the noise of 3 that will be added to the cell count 362 using the ckey 0.1862791?
  - 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
- **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
# 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
```

```
## 1: 0 0 0.00 1 1.0000 0
## 2: 1 0 1.05 1 0.0000 1
## 3: 2 0 1.05 1 0.5557 1
## 4: 3 0 1.05 1 0.2881 6
## 5: 4 0 1.05 1 0.5000 1
```
### Example (continued)

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

```r
## i  j    p   v p_int_lb p_int_ub type
## 1  0  0 1.000000 0  0.000000 1.000000 all
## 2  1  0 0.508333 -1 0.000000 0.508333 all
## 3  1  2 0.475000  1 0.508333 0.983333 all
## 4  2  0 0.161558 -2 0.983333 1.000000 all
## 5  2  0 0.161558 -2 0.000000 0.161558 all
## 6  2  2 0.555650  0 0.161558 0.717209 all
```
Example (continued)

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

## Perturbation Panel

![Diagram of perturbation panel](image_url)

- **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** 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:

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

<table>
<thead>
<tr>
<th>level</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>@</td>
</tr>
<tr>
<td>2</td>
<td>@@</td>
</tr>
<tr>
<td>3</td>
<td>@@@</td>
</tr>
<tr>
<td>4</td>
<td>@@@</td>
</tr>
<tr>
<td>5</td>
<td>@@</td>
</tr>
<tr>
<td>6</td>
<td>@@@</td>
</tr>
<tr>
<td>7</td>
<td>@@@</td>
</tr>
<tr>
<td>8</td>
<td>@@</td>
</tr>
</tbody>
</table>
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 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
```
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.

## Count variable 'high_inc' was perturbed.

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

<table>
<thead>
<tr>
<th></th>
<th>sex</th>
<th>age</th>
<th>vname</th>
<th>uwc</th>
<th>wc</th>
<th>puwc</th>
<th>pwc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>total</td>
<td>total</td>
<td>high_inc</td>
<td>445</td>
<td>26535</td>
<td>444</td>
<td>26475.3708</td>
</tr>
<tr>
<td>2</td>
<td>total</td>
<td>ag1</td>
<td>high_inc</td>
<td>192</td>
<td>11881</td>
<td>190</td>
<td>11757.2396</td>
</tr>
<tr>
<td>3</td>
<td>total</td>
<td>ag2</td>
<td>high_inc</td>
<td>123</td>
<td>6833</td>
<td>122</td>
<td>6777.4472</td>
</tr>
<tr>
<td>4</td>
<td>total</td>
<td>ag3</td>
<td>high_inc</td>
<td>82</td>
<td>4980</td>
<td>78</td>
<td>4737.0732</td>
</tr>
<tr>
<td>5</td>
<td>total</td>
<td>ag4</td>
<td>high_inc</td>
<td>34</td>
<td>1967</td>
<td>33</td>
<td>1909.1471</td>
</tr>
<tr>
<td>6</td>
<td>total</td>
<td>ag5</td>
<td>high_inc</td>
<td>14</td>
<td>874</td>
<td>12</td>
<td>749.1429</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>sex</th>
<th>age</th>
<th>vname</th>
<th>uws</th>
<th>ws</th>
<th>pws</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>total</td>
<td>total</td>
<td>savings</td>
<td>2273532</td>
<td>136526777</td>
<td>136526792</td>
</tr>
<tr>
<td>2</td>
<td>total</td>
<td>ag1</td>
<td>savings</td>
<td>982386</td>
<td>59675958</td>
<td>59674846</td>
</tr>
<tr>
<td>3</td>
<td>total</td>
<td>ag2</td>
<td>savings</td>
<td>552336</td>
<td>32920428</td>
<td>32917962</td>
</tr>
<tr>
<td>4</td>
<td>total</td>
<td>ag3</td>
<td>savings</td>
<td>437101</td>
<td>26182380</td>
<td>26179921</td>
</tr>
<tr>
<td>5</td>
<td>total</td>
<td>ag4</td>
<td>savings</td>
<td>214661</td>
<td>12382655</td>
<td>12383491</td>
</tr>
<tr>
<td>6</td>
<td>total</td>
<td>ag5</td>
<td>savings</td>
<td>80451</td>
<td>4963278</td>
<td>4954843</td>
</tr>
</tbody>
</table>
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/](sdctools.github.io/cellKey/)
- **Questions**: [github.com/sdcTools/userSupport](github.com/sdcTools/userSupport)
- **Bug-Reports**: [github.com/sdcTools/userSupport/issues](github.com/sdcTools/userSupport/issues)
Consistently perturbing tables
Introducing R package cellKey