Testing CTA as Additivity Module for Perturbed Census 2021 EU Hypercube Data

Tobias Enderle[*],Sarah Giessing[**]

[*] Statistisches Bundesamt, 65180 Wiesbaden, Germany, Tobias.Enderle@destatis.de
[**] Statistisches Bundesamt, 65180 Wiesbaden, Germany, Sarah.Giessing@destatis.de

Abstract: The EU project “Harmonized Protection of Census Data in the ESS” aims at testing disclosure control methods that might be suitable for protection of Census data in the ESS for the Census 2021 round. In the project, it was decided to consider two different perturbative approaches, e.g. record swapping and cell key based random noise, a.k.a the ABS method. Cell key based random noise is a post-tabular method which can be implemented in two steps. In the first step, independent random noise is added to the original count data. One effect is that the perturbed hypercube data will generally not add up exactly. The Australian Statistics Bureau (ABS) is therefore using the method along with an “additivity module”. Notably, non-additivity or a need to restore additivity is no issue with methods like record swapping that are applied on the level of microdata.

This paper looks into the feasibility of using the controlled tabular adjustment algorithm (CTA) as tool for an additivity module. We report the results of testing the approach on a synthetic data set with the data structure of one of the 4-dimensional Census 2021 hypercube selected for the testing in the project. We will also evaluate the performance of the approach and compare the effect of the additivity adjustment to the noise design effects.

1 Introduction

After the first step, a fundamental property of cell key based random noise is its *consistency*, i.e. the random noise added to a specific cell will always be exactly the same, even if the cell appears in e.g. different hypercubes. Random noise can be designed with certain desirable properties like unbiasedness, constant noise variance, fixed maximum deviation between original and noisy counts and ensuring that certain frequencies (viz., 1s and 2s) will not appear in the perturbed data. It is up to the disseminator to choose a suitable parameter for noise variance and maximum deviation (Fraser and Wooton, 2006; Antal, Enderle and Giessing, 2017; Giessing, 2016).

To some degree restoring additivity will spoil the design properties of the noise and also the consistency property. In this paper, we study options and limitations of restoring additivity while still maintaining consistency within Census hypercubes, or even between Census hypercubes. In this study we use the controlled tabular adjustment algorithm (CTA; Castro et al., 2013) that is, for example, available in the τ-ARGUS package as tool for an “additivity module”.

In Section 2 we briefly recall the CTA algorithm. Section 3 discusses various strategies and options of using the algorithm, some preliminary test results are presented in Section 4. The paper finishes with a summary and some conclusions.

2 CTA Algorithm as Additivity Module

The procedures to be outlined and discussion in section 3 below rely on the CTA package. In the present section we briefly summarize the methodology and some important options implemented in the package.

Leaning to the denotation of Castro et al. (2013), a CTA instance is represented by (i) a set of cells y_i; $i = 1,\ldots,n$, that satisfy m linear relations $Ay = b$ (y being the vector of y_i's; matrix A and vector b imposing the tabular constraints, expressing for example that the cell values of some set of cells must be identical to the cell value of another (marginal) cell); (ii) a lower and upper *a priori* bound for each cell $i = 1,\ldots,n$, respectively l_i and u_i , which can be used to impose that the adjusted values are still "similar" to the data before adjustment. In addition to that we can also define (iii) a set $P = \{i_1, i_2, \ldots, i_p\} \subseteq \{1,\ldots,n\}$ of indices of "sensitive cells" and require (iv) for each sensitive cell $i \in P$ a lower and upper protection level, respectively lpl_i and upl_i, such that the adjusted values satisfy either $x_i \geq y_i + upl_i$ or $x_i \leq y_i - lpl_i$.

In the context of restoring additivity to a perturbed hypercube, constraint (i) defines the additive relations in the hypercube to be restored eventually. Constraints (ii) can be used to control the effect of the adjustment on the data. Notably, by means of (iv) we can define for specific "sensitive" cells, defined by (iii), intervals $[y_i - lpl_i ; y_i + upl_i]$ that *must not* contain the adjusted cell value. In the context of adjusting perturbed tables of counts, this can be useful when certain counts (*viz.*, 1s and 2s) should not appear in the data after adjustment.

Given these settings of the CTA instance, the purpose of CTA is to find the set of closest feasible adjusted values x_i; $i = 1,\ldots,n$ satisfying these conditions. This is expressed as the following optimization problem (in terms of the deviations $z_i =: x_i - y_i$ and w_i being a vector of cell weights)[1]:

$$(1) \qquad \begin{aligned} &\min_z \sum_{i=1}^{n} w_i \, | z_i | \\ &s.t. \quad Az = 0 \\ &\qquad l_i \leq z_i \leq u_i \quad , i = 1,\ldots, n \\ &\qquad z_i \leq -lpl_i \; or \; z_i \geq upl_i , \quad i \in P \end{aligned}$$

A challenge comes from the last line of this statement relating to the constraint set (iv), because of the "or" condition. The exact formal mathematical expression of this condition requires the introduction of a vector of binary variables into the problem, associated to the direction of the deviation of each "sensitive" cell, i.e. cell with an interval that should exclude the adjusted value. An optimal solution requires an optimal allocation of these "directions" to the binary variables and makes the CTA

[1] For the exact mathematical statement and the linearity issue of the optimization problem see Castro et al. (2013).

problem a computationally “difficult” mixed integer linear programming problem (MILP). It can be relaxed significantly by turning it into a continuous (convex) optimization problem which is computationally much easier to solve.

3 Strategies and Options for using CTA as Additivity Modul for Perturbed Census Hypercube Data

The purpose of the strategies described in this paper is making each individual EU hypercube additive. As mentioned in the introduction, this can spoil consistency of the figures between hypercubes. Because hypercubes are large, it will also usually not be feasible to restore additivity to a hypercube in a single, simultaneous CTA step, not even in the “simple” case where CTA is a “simple” convex optimization problem. Hence, we develop heuristic blocking strategies that on the one hand finally result in fully additive hypercubes and should on the other hand keep inconsistencies between different hypercubes as small as possible. Keeping this in mind, for this paper we further simplify the problem by focussing on only one of the hypercubes considered in the project, e.g. hypercube 9.2[2], defined as cross combination of the variables *GEO.M* (Geography), *SEX*, *AGE.M*, and *YAE.H* (Year of arrival in the country since 1980)[3]. Nevertheless, the principles of the methodology devised for this particular hypercube should be “generic” in the sense that they are expected to be easily adaptable also to the other hypercubes (e.g. EU Hypercube 9.1 or 9.3).

We will also distinguish two different types of applications, i.e. the *“simple”* and the *“complex”* case. We are in the simple case, if the disseminator does not require that certain (small) counts (i.e. 1’s, 2’s, …) do not appear in the hypercubes after adjustment. In that case, the set P of “sensitive” cells is empty. Hence, the last line in (1) can be omitted which turns the problem defined by (1) into a “relatively” easy to solve convex optimization problem.

The complex case occurs, when the adjusted data shall indeed not contain small counts. In that case, we assume that in the perturbation step noise distribution parameters have been used to enforce that the perturbed data do not contain any such counts. However, it can happen that such counts show up again in the solution obtained by CTA. This could be prevented in a straightforward way by a rigorous definition of the type (ii) constraints. However, such straightforward techniques bear a risk of defining infeasible problems as illustrated in Example A.1 (Appendix A.1).

[2] Details on the content of the hypercubes can be found in the Census 2021 draft implementing regulation that was approved at the 30th Meeting of the European Statistical System Committee on 28 September 2016 (item 2 of the agenda). See Eurostat Unit F2 (2016).

[3] Some variables have more than one breakdown, each with different levels of detail. In the terminology of the draft implementing regulation, ‘H’ identifies breakdowns with the highest level of detail, ‘M’ identifies breakdowns with a medium level of detail, and ‘L’ identifies breakdowns with the lowest level of detail and ‘N’ identifies the breakdown that refers to the national level. See Table A.1 (Appendix A.2) for the number of categories per variable of our test-hypercube 9.2.

3.1 Blocking strategy

In the following, we enumerate the variables of a Census hypercube using roman numbers. Thus, (y_I, A_I, b_I) denotes a one-dimensional sub-instance involving only the first variable of an instance hypercube. The 2-dimensional sub-instance (y_{II}, A_{II}, b_{II}) would involve all two way cross-combinations of the first two variables including the respective margins. Analogously, we define higher dimensional sub-instances up to the complete instance (y, A, b). This is referred to as **vertical split** of the blocking.

Another dimension of the blocking is given by a **horizontal split**. We use higher aggregate levels of one or more variables, e.g. geography for this horizontal blocking, enumerating $k = 1, \dots, K$ blocks splitting the original hypercube (y, A, b) into K separate pieces $((y, A, b)_k)_{\ k=1,\dots K}$. Each sub-hypercube $(y, A, b)_k$ contains data for the corresponding population sub-group k only. However, each sub-hypercube is of full dimension and provides counts of sub-population k at full detail. Since the sub-populations are treated separately, we must sum across the K sub-hypercubes (also referred to as "hierarchical" aggegation) in order to obtain full population margins. Alternatively, it might be decided that it is adequate just to ensure additivity within each of the K blocks. In the latter case one should foresee separate blocks above and below the hierarchy level of the splitting variable.

Appendices A.2 and A.3 give a more detailed description of the used four-dimensional hypercube 9.2, its variables and the splitting strategy we applied.

In order to keep better control of the margins (i.e., their deviations from the perturbed as well as original data) and, hence, from "identical" cells in linked hypercubes, we introduce a three step procedure:

***Step 1*:** Apply CTA to a sub-instance resulting from a vertical split of the full target hypercube, for example to sub-instance $(y_{III}, A_{III}, b_{III})$ (population by geography, age and sex). Note that many Census hypercubes include these three variables (e.g. EU Hypercubes 9.3 and 9.4). So re-using the outcome of this step should help to better preserve consistency between these hypercubes.

Step 2: Apply CTA to the low-detail level sub-hypercube $(y, A, b)_L$ that presents counts of population by geography (*GEO.L*), age (*AGE.L*), *SEX* and year of arrival in the country (*YAE.L*).

***Step 3*:** Apply CTA separately to the K blocks $(y, A, b)_k$ resulting from a horizontal split.

The three steps should of course not be independent. When preparing instances for step 2 and 3, we take into account the results of the previous steps.

Therefore we now introduce an extended denotation. Instead of x, y and z, in the following, we denote $y^{(t-1)}$ the vector of the data which is input of a CTA execution with index t, $z^{(t)}$ the vector of deviations obtained by this execution and $y^{(t)} =$

$y^{(t-1)} - z^{(t)}$ the data vector after adjustment in CTA step t. A more detailed description of the steps is given in Appendix A.3.

3.2 Strategies for the complex case

In the case we refer to as "complex" case, we require all entries of a "feasible" solution vector y to be either zero or above a given threshold N (N=2, typically). It is assumed that a suitable choice of parameters for the cell key method ensures that the initial dataset satisfies this requirement.

3.2.1 Direct approach

To express the complex case as CTA instance directly, we define a set P of sensitive cells actually depending on the lower *a priori* bounds l_i: a cell with index i is sensitive, if $y_i - l_i \leq N$. For those sensitive cells $i \in P$ we define lower and upper protection levels $lpl_i := y_i$ and $upl_i := N + 1 - y_i$. Hence, in a feasible solution we have $x_i \geq y_i + upl_i = y_i + N + 1 - y_i$=N+1 or $x_i \leq y_i - lpl_i = y_i - y_i = 0$. Notably, this is different from the "usual" way of using CTA. Usually, when a cell is declared sensitive to CTA, protection levels are defined enforcing that its value is changed in a feasible solution. In our case, sensitive cells may keep their value – they are only not allowed to change in certain way.

Since the CTA routine can't handle negative protection levels so far, we applied a workaround and shift those sensitive cells where y_i > N+1 downwards to $y_{i,shifted}$ = N+1 and, respectively, redefine lower and upper protection levels $lpl_{i,shifted} := y_{i,shifted}$ and $\mathrm{upl}_{i,\mathrm{shifted}}$:= 0. The initial lower and upper bounds remain unaffected and are still related to y_i rather than $y_{i,shifted}$.

However, due to their size and the high detail of EU Hyperubes this kind of setup may lead to instances with many "smaller" cells to be declared sensitive. This may increase the complexity of the problem and finally require splitting the data in a very large number of small horizontal blocks to keep computing requirements manageable. But more splitting will tend to result in less optimal solutions. The next subsection discusses an alternative strategy.

3.2.2 Alternative strategy

The heuristic approach discussed in the following is expected to require more than one execution of CTA on the same hypercube (or block of a hypercube). Although more elaborate strategies with more iterations can be imagined. To keep it simple we describe here a short process with only one execution of the second phase:

Initial Phase: Like in the simple case, no cells are declared sensitive in CTA problems defined in the initial phase (t=1). Even though we assign high weights to "smaller" cells (i.e. those to be declared sensitive in the above direct approach), some cells $y_i^{(0)}$ may nevertheless be adjusted to $y_i^{(1)}$, where 0 < $y_i^{(1)} \leq$ N.

Second Phase: For CTA execution with index *t=2*, define as set of sensitive cells $P^{(2)} := \{i; 0 < y_i^{(1)} \leq \mathrm{N}\}$. For $i \in P^{(2)}$ define protection levels $lpl_i := y_i^{(1)}$ and $upl_i := N + 1 - y_i^{(1)}$. Then, in a feasible solution $y_i^{(2)} > N$ or $y_i^{(2)} = 0$ for all $i \in P^{(2)}$.

In order to avoid that some other cells ($i \notin P^{(2)}$) are adjusted in this phase to $y_i^{(2)}$, where $0 < y_i^{(2)} \leq N_,$ we define special constraints. However, to avoid infeasibility, these special constraints shall be defined only for cells that do not appear as margin in any of the hypercube relations defined by matrix A. Let B be the subset of these "bottom level" cells.

Define lower *a priori* bounds $l_i < y_i^{(1)} - N$, for all non-zero cells in B but not in $P^{(2)}$, i.e. where $y_i^{(1)} > N$. For all zero-cells in B, i.e. where $y_i^{(1)} = 0$, define upper *a priori* bounds $u_i = 0$.

Defining the special upper and lower bounds only on the subset of bottom level cells *B* avoids infeasibility problems like the one illustrated by Example A.1. And, due the structure of the EU Census hypercubes, it is sufficient to enforce (by the special constraints) only for cells in B that the solution vector $y_i^{(2)}$ does not contain any non-zero entry $< N+1$, because by definition of the Census hypercubes, all higher level cells, i.e. the cells not in B, in a feasible (additive) solution $y_i^{(2)}$ are summations of cells in B. Obviously, summing counts that are either zero, or greater than N can never result in a positive count $\leq N$.

Final Phase: Nevertheless, the second phase might still lead to a CTA result with some $0 < y_i^{(2)} \leq N_,$ for example due to too tight *a priori* bounds. To those cells, if they are bottom level cells, i.e. $i \in B$, we now apply simple deterministic rounding of sensitive counts to the basis 3 (i.e., changing them all into 0 or 3). Finally we derive an additive hypercube by obtaining results for cells $i \notin B$ simply through aggregation across cells in B.

3.3 Use of weights

Important for influencing the behaviour of a CTA procedure is the setting of weights. For the testing reported in this paper, we follow a suggestion of Castro and Giessing (2006) and assign weights of the form $y_i^{-\gamma}$, $0 \leq \gamma \leq 1$. Small choices of γ make all weights rather similar, while with larger γ, smaller cells get relatively high costs, and will thus be better preserved at the expense of some more perturbation in larger (perhaps margin) cells. For the first, preliminary testing reported in this paper we decided to use $\gamma := 0.5$.

With the blocking strategy of Sec. 3.1 we indirectly increase weights for those cells where $y_i^{(t)} \neq y_i^{(t-1)}$(due to the replacement of $y_i^{(t-1)}$ by an entry from the solution vector of a previous CTA execution on a more aggregated part of the hypercube) in order to better preserve the solutions of the previous CTA execution and hence consistency across different blocks. Actually this was done indirectly, by decreasing the weights of the remaining cells which were given weights of the form $y_i^{-\gamma}/1000$.

For the initial CTA execution described for the strategy of Sec. 3.2.2 we also need an extended weighting scheme. Here we assign big weights to "smaller" cells to avoid that they are selected for adjustment because they might then be adjusted "in the wrong way", turning into a non-zero count less than N+1which would be considered sensitive in the next phase. A simple approach taken in the testing reported in the next section is based on the fact that except for the second phase of the strategy of Sec. 3.2.2 we always define lower *a priori* bounds l_i as min(*D*; y_i) with a suitable constant parameter *D* (f.i. *D*=10). The weighting scheme is of the form $b^{\lambda(y_i)}y_i^{-\gamma}$, with a suitable basis b (like f.i. b=2) raised to the power of $\lambda(y_i)$, where $\lambda(y_i) := D,$ for $y_i = 0,\ \lambda(y_i) := \frac{D}{y_i - 2},$ for $2 < y_i \leq D + 2,$ and $\lambda(y_i) := 0,$ otherwise.

4 Testing the proposed strategies

For testing of the proposed strategies we use a synthetic data set with distributions similar to those of the Census 2011 data for some parts of the north of Germany (cf. see Tables A.1 and A.2). Some of the testing is still on-going, so results have a preliminary character. Also, parts of the strategies proposed in Sec. 3 have been conceived to overcome disadvantages that became visible when looking at test results obtained with strategies devised earlier. Using the cell key method we applied noise to our test hypercube 9.2 using two variants for the noise. In the first setting, we use the variant recommended in Antal, Enderle & Giessing (2017) where the noise variance is 1 and the maximum deviation is 3. For testing the complex case we use a noise variant with maximum deviation 3 and variance 2 where small counts of 1 and 2 do not appear in the set of perturbed cell values.

4.1 Testing the blocking strategy in the "simple case"

Before testing the blocking strategy we tried to run the fully specified CTA instance (y, A, b) of the 4-dimensional hypercube inclusiding all hierarchical levels. However, we manually stopped the CTA run after one week due to size and complexity of the hypercube by means of numbers of cells and equations. Table A.3 (Appendix A.3) gives an overview of this CTA instance as well as the instances we ran for testing the blocking strategy.

In particular, we only present results for a combination of steps 1 and 3, i.e. omitting step 2. The idea of an additional step 2 as described in Sec. 3.1 turned up only after inspecting results. It has not yet been tested.

As step 1 sub-instance $(y_{III}, A_{III}, b_{III})$ we use the hypercube *GEO.M x AGE.M x SEX*. For the blocking instances $((y, A, b)_k)$ $_{k=1,\ldots K}$ within step 3 we split the original hypercube *GEO.M x AGE.M x SEX x YEA.H* into K=6 blocks, using the second aggregate level of the *GEO.M* variable. Thus, each block (i.e., geo1 to geo6) represents the hypercube data of one NUTS-2 region.

The CTA settings - such as *a priori* upper and lower bounds (i.e. maximum allowed deviations) or optimal solution gaps - are set in accordance with size and complexity of the corresponding region k. Table A.3 gives an overview of sizes, settings, resulting CPU times and maximum deviations. In three out of six regional blocks the initially set boundaries are marginally violated.

The counts of overlapping cells of the step-wise procedure that got adjusted within the step 1 sub-instance are carried over to the respective K instance blocks of step 3. The weights of the non-overlapping and, hence, still unadjusted cells are – as described in Sec. 3 - shrinked additionally by the factor 1/1000, such that the overlapping cells will come along with much higher costs of adjustment within step 2. Table A.4 (Appendix A.3) shows the efficiency of this approach as exemplified by differences of the overlapping cells within the NUTS-2 region geo4. The step-wise approach results in a maximum absolute deviation of 1, whereas an independent run of the identical CTA instance produces deviations up to 6.

Since YAE.H was omitted in the first step and the blocks in step 3 are constructed by splitting the regions on the NUTS 2 level, higher aggregates of *GEO.M* (i.e. NUTS 1 and the total population level) regarding the fourth dimension YAE.H are neither adjusted nor additive yet. In a further so-called aggregation step, the adjusted cells of the blocks within step 3 must be summed up to NUTS 1 aggregates which again must be summed up afterwards to the total population aggregates. However, such a “simple” hierarchical aggregation may lead to large deviations as can be seen in Table A.5 (Appendix A.3). While all deviations up to step 3 are within the selected CTA boundaries of maximum 10, the hierarchical aggregation causes a maximum absolute deviation of 17.

An improvement might be possible by including step 2 based on the low-detailed 4-dimensional hypercube *GEO.L x AGE.L x SEX x YEA.L* and, once again, adjusted weights in the process. Hence, an adjustment of higher aggregates (e.g. NUTS 1 level) regarding all dimensions before running the blocking instances may reduce the maximum absolute deviations of these higher aggregates as we have learned from the comparison in Table A.4. This may also reduce the inconsistency issue.

4.2 Testing “complex case” strategies

So far, we tested the strategies for the complex case only on a few instances, not on the entire hypercube.

The direct approach described in Sec. 3.2.1 resulted in an infeasible solution due to the additivity constraints. Several linear relations of the cells are violated and can’t be solved given the remaining constraints, especially the protection levels of sensitive cells.

For testing the alternative strategy of Sec. 3.2.2, we use the 4-dimensional hypercube *GEO.M x AGE x SEX x YAE*[4] where *AGE* is (a) *AGE.L* for first experiments on smaller instances and (b) *AGE.M* to look at a bigger instance.

(a) Small instance: A first lesson learnt from trying the initial execution phase on the smaller instance is that the concept of only increasing the weight for the smaller cells does not help (cf. Table A.7, Appendix A.4): without increasing the weights, in this instance 25 cells were adjusted to a (sensitive) count of 2. When running the instance again with extended weights of the suggested form $b^{\lambda(y_i)}y_i^{-\gamma}$, still 24 cells turn into 2s. A more sophisticated strategy by setting more restrictive *a priori* bounds (lower bounds are set to $\min\big(\mathrm{D}; \max(y_i - (N+1); 0)\big)$ for bottom level cells that are non-sensitive[5]) works better: it leads to 12 cells with adjusted counts 1 or 2 in an application without the special weighting scheme. Testing the stricter lower bounds in combination with the weighting scheme, the CTA solution reduces the number of remaining sensitive cells to 4 (this time only counts of 1).

(b) Big instance: For testing also the second phase of the procedure described in Sec. 3.2.2, we used the bigger instance with hypercube *GEO.M x AGE.M x SEX x YAE*. After the initial CTA execution using extended weights and the stricter lower bounds for non-sensitive bottom level cells there are 167 sensitive cells (cf. Table A.7). Table A.8 (Appendix A.4) shows the frequencies after the iterations of the second phase until there is no further improvement with regard to the number of sensitive cells. The iterative procedure can almost help to avoid sensitive cells. The CPU time of the additional iterations 2 to 5 is negligible in relation to the time of the initial phase. After the final phase of rounding to the base of 3 and the computation of higher aggregates using only bottom level cells, the hypercube is additive and has a maximum absolute deviation of 16. In contrast to this, using a straight-forward hierarchical aggregation of the perturbed cell frequencies obtained from the random noise step (i.e., the simplest approach to get an additive hypercube) we receive much higher absolute deviations up to 175 (cf. Table A.9, Appendix A.4) and a more serious inconsistency issue.

5 Summary and Conclusions

This paper has presented a variety of heuristics how to use the CTA algorithm for restoring additivity to EU Census hypercubes after an initial perturbation using the cell key method, also studying a complex case, where certain low counts would be considered sensitive and should not appear in the adjusted hypercube.

Without these complicating constraints the CTA problem is a “simple” convex optimization problem. However, due to size and complexity of the hypercubes, at least

[4] To make testing the complex case strategies feasible, we applied an additional breakdown definition of year of arrival with no hierarchy. YAE is only on the highest aggregation level (i.e., 1., 2. and 3.).

[5] The non-bottom level cells as well as sensitive cells will receive a lower bound of $\min(D; y_i)$.

on a “standard” PC we did not succeed to restore additivity to one complete EU Census hypercube within a single CTA step – not to speak of simultaneous adjustment of the entire set of more than 70 hypercubes. On the other hand, using the heuristic blocking approach proposed in the paper, the problem could be solved within reasonable time for our test instance, similar in size to one hypercube for one smaller EU member state (like, say, Slovenia).

An important (though trivial) finding is that an approach like CTA which “balances” the adjustments within the hypercube relations leads to much less perturbation in hypercube margins than simple summation of the noisy lowest level counts. Also not surprising is the result that original counts are best preserved without additivity adjustment, and that information loss due to restoring additivity is generally the smaller, the smaller the “instance” considered for adjustment. As a conclusion, we recommend either not to restore additivity at all, or to restore additivity to sub-hypercubes only. A good compromise might be to produce a separate, additive hypercube per geographic area foreseen in the full hypercube, not requiring that the summed adjusted counts of lower level geographic areas match exactly the adjusted figures in the (additive version of the) sub-hypercube relating to the respective geographic region on the higher level of the geography classification (NUTS). The paper has proposed techniques which should help to preserve to some degree consistency of adjusted counts across hypercubes at least for low level of detail topic breakdown.

For the complex case with additional constraints some preliminary encouraging results were obtained, but not yet on fully detailed sub-hypercubes. Member states where disclosure limitation policy requires such constraints and exact additivity of the hypercubes is considered highly desirable might wish to consider using a method like SAFE (cf. Höhne, 2011) which produces an “optimally” perturbed micro-data set in which all possible variable combinations appear at least 3 times or not at all. However, this alternative may come along at the expense of higher information loss (i.e., higher variances of perturbations).

For future work it might be interesting also to consider a calibration approach based on an iterative algorithm such as iterative proportional fitting and compare it to CTA.

Acknowledgements. The research leading to these results has partially received funding from the EU project “Harmonized Protection of Census Data in the ESS” (Specific grant agreement n° 11111.2016.005-2016.367 under the Framework partnership agreement n° 11112.2014.005-2014.533).

References

Antal, L., Enderle, T. & Giessing, S. (2017). '*Statistical disclosure control methods for harmonised protection of census data*', Deliverable D3.1 of Work Package 3 'Development and testing of recommendations; identification of best practices' within the Specific Grant Agreement 'Harmonised protection of census data in the ESS'.

Castro, J. & Giessing, S. (2006). *Quality issues of minimum distance controlled tabular adjustment.* Paper presented at the European Conference on Quality in Survey Statistics (Q2006), 24.-26. April 2006 in Cardiff.

Castro, J., Gonzalez, J.A., Banea, D & Jimenez, X. (2013). *User's and programmer's manual of the RCTA package (v.2).* Technical Report DR 2013-06, available from: http://www-eio.upc.es/~jcastro

Eurostat Unit F2 (2016): *Commission implementing Regulation laying down rules for the application of Regulation (EC) No 763/2008 of the European Parliament and of the Council on population and housing censuses as regards the technical specifications of the topics and of their breakdowns*, Item 2 of the agenda, 30th Meeting of the European Statistical System Committee, 28th September 2016, ESSC 2016/30/3/EN, 28.09.2016.

Fraser, B. and Wooton, J. (2006). *A proposed method for confidentialising tabular output to protect against differencing*. In: Monographs of Official Statistics. Work session on Statistical Data Confidentiality, Eurostat-Office for Official Publications of the European Communities, Luxembourg, 2006, pp. 299-302.

Giessing, S. (2014), '*Pre-tabular Perturbation with Controlled tabular Adjustment: Some Considerations'*. In: Domingo-Ferrer, J. (Eds.), Privacy in Statistical Databases, pp. 48-61, Springer, LNCS, vol. 8744.

Giessing, S. (2016), '*Computational Issues in the Design of Transition Probabilities and Disclosure Risk Estimation for Additive Noise'.* In: Domingo-Ferrer, J. and Pejić-Bach, M. (Eds.), Privacy in Statistical Databases, pp. 237-251, Springer International Publishing, LNCS, vol. 9867.

Höhne, J. (2011), '*SAFE – A method for anonymising the German Census'.* Paper presented at the Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality (Tarragona, 26-28 October 2011) available at https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2011/16_Germany.pdf

Appendix A.1

Example A.1. Consider a table consisting of a single line, presenting noisy counts of inhabitants with a certain rare property x in total and by sex

	male	female	total
Property x	0	0	3

Assume, no counts of 1 or 2 are allowed in the table after adjustment. We might be tempted to define as type (ii) constraints for both, the males- and the females-cell upper bounds of 0 for the CTA deviation to avoid that CTA turns them into a 1 or a 2. We might also define a lower bound of 0 for the CTA deviation of the total-cell, again to avoid that CTA turns it into a 1 or a 2. Obviously, all three conditions together turn the CTA instance infeasible.

Appendix A.2 – Test hypercube 9.2

Table A.1 gives an overview of the hypercube 9.2 we used for the testing. The sub-instance (y_I, A_I, b_I) would offer population counts by *GEO.M*, e.g. down to NUTS 3 level. The 2-dimensional sub-instance (y_{II}, A_{II}, b_{II}) would involve all two way cross-combinations of the first two variables (including the respective margins), i.e. the (y_{II}, A_{II}, b_{II}) sub-hypercube presents population counts by *GEO.M* and *AGE.M* (i.e., down to 5-year age band). Analogously, we define the 3-dimensional sub-instance $(y_{III}, A_{III}, b_{III})$ involving the population by geography, age and sex.

Dimension	**I**	**II**	**III**	**IV**
Variable	GEO.M	AGE.M	SEX	YAE.H
Number of categories (bottom level cells)	42	21	2	24
Hierarchical structure	Three geographical area levels from NUTS 1 to NUTS 3	Two age levels	No levels	Two year of arrival levels

Table A.1. Variable Description and Hierarchical Structure of the test hypercube 9.2 (number of hierarchical levels below the total population). The geography is separated in three NUTS 1 regions. The first has one NUTS 2 region and 11 NUTS 3 areas; the third has four NUTS 2 regions each with 8, 6, 12 and 6 NUTS 3 areas. The second NUTS 1 region has no further breakdowns.

Appendix A.3 – Simple Case

The three step procedure we suggest for the blocking strategy would read as follows:

Step 1: The variables of the first CTA step on the sub-instance $(y_{III}, A_{III}, b_{III})$ are *GEO.M x AGE.M x SEX*. Assuming t=1, we receive the sub-instance and the resulting output vector

$$(y_{III}, A_{III}, b_{III}) \overset{\text{def}}{=} \left(y_{III}^{(0)}, A_{III}, b_{III}\right) \text{ and } y_{III}^{(1)} = y_{III}^{(0)} - z_{III}^{(1)}$$

Step 2: We denote the CTA input vector y of the step 2 sub-instance $(y, A, b)_L$ by $y^{(1)}$. Notably, all cells in $y^{(1)}$ for up-to-3-way margins not involving the 4th variable (i.e. where *YAE* category refers to the population total) "inherit" their entries from $y_{III}^{(1)}$. For any of the other cells, entries of $y^{(1)}$ are still those from the respective entries of $y^{(0)}$. Thus, the sub-instance and the resulting output vector could be defined as

$$(y, A, b)_L \overset{\text{def}}{=} \left(\left(y_k^{(1)} \middle| y_{III}^{(1)}, y^{(0)}\right), A, b\right)_L$$

and

$$y_L^{(2)} = \left(y_L^{(1)} \middle| y_{III}^{(1)}, y^{(0)}\right) - z_L^{(2)}$$

Step 3: The input vector for block k is then $y_k^{(2)}$. In each of the *K* blocks, $y^{(2)}$ has entries from $y_{III}^{(2)}$ only for the up-to-3-way margins. The other cells of $y_k^{(2)}$, e.g. those presenting frequencies by year of arrival in the country, are either still those from the respective entries of $y^{(0)}$, or those of $y^{(2)}$ (for cells corresponding to *GEO.L x AGE.L x SEX x YEA.L* margins or sub-margins). Thus, the sub-instance and the resulting output vector could be defined as

$$(y, A, b)_k \overset{\text{def}}{=} \left(\left(y_k^{(2)} \middle| y_L^{(2)}, y_{III}^{(2)}, y^{(0)}\right), A, b\right)_k$$

and

$$y_k^{(3)} = \left(y_k^{(2)} \middle| y_L^{(2)}, y_{III}^{(2)}, y^{(0)}\right) - z_k^{(3)}$$

However, for the testing we omitted step 2 as described in Sec. 4.1. Thus, the blocking strategy we applied for the testing can be reduced as follows:

Step 1: $(y_{III}, A_{III}, b_{III})$ $\quad y_{III}^{(1)} = y_{III}^{(0)} - z_{III}^{(1)}$

Step 3: $\left(\left(y_k^{(1)} \middle| y_{III}^{(1)}, y^{(0)}\right), A, b\right)_k$ $\quad y_k^{(3)} = \left(y_k^{(1)} \middle| y_{III}^{(1)}, y^{(0)}\right) - z_k^{(3)}$

CTA Instance	**Step t**	**Number of cells**	**Number of equations**	**CPU time** (in seconds)	**CTA bound**	**CTA gap** (%)	**Maximum absolute deviation**
(y, A, b)	-	133,560	129,822	-	15	5	-
$(y_{III}, A_{III}, b_{III})$	1	4,452	3,437	9	1	5	3
$(y, A, b)_{geo1}$	3	30,240	26,208	45,099	5	5	10
$(y, A, b)_{geo2}$	3	5,040	6,468	9	5	5	5
$(y, A, b)_{geo3}$	3	22,680	20,268	16,932	10	5	10
$(y, A, b)_{geo4}$	3	17,640	16,338	15,817	5	10	10
$(y, A, b)_{geo5}$	3	30,240	26,208	164,848	10	5	8
$(y, A, b)_{geo6}$	3	17,640	16,338	22,527	5	5	8

Table A.3. CTA settings, CPU times and maximum absolute deviations. All step 3 instances we applied depend on the previous step 1 instance $(y_{III}, A_{III}, b_{III})$. For the computation we used a standard Desktop PC with Intel CPU (i5-2500S @2.70Ghz) and 8 GB of RAM. We manually stopped the CTA run of the instance (y, A, b) after one week.

$\|\boldsymbol{z}\|$	**Frequencies of deviations between noisy cell counts before and after CTA observed for overlapping cells after step 2 instance $(\boldsymbol{y}, \boldsymbol{A}, \boldsymbol{b})_{\boldsymbol{geo4}}$ …**	
	conditioned on step 1 sub-instance $(\boldsymbol{y}_{\boldsymbol{III}}, \boldsymbol{A}_{\boldsymbol{III}}, \boldsymbol{b}_{\boldsymbol{III}})$	**independently**
0	576	170
1	12	191
2		139
3		57
4		22
5		8
6		1
All	588	588

Table A.4. Comparison of the step-wise approach and an independent execution of step 2 regarding the overlapping cells: frequencies of the deviations between noisy cell counts before and after CTA.

|z|	**Frequencies of deviations between original counts and counts after …**		**Frequencies of deviations between noisy counts before adjustment and after CTA or aggregation steps …**			
	Random Noise (non-additive)	**CTA** (additive)	**Step 1**	**Step 3**	**Hierarchical Aggregation**	**Step 3 and hierarchical aggregation**
0	56,708	57,846	1,373	50,216	2,431	52,647
1	61,982	58,449	2,951	57,568	3,267	60,835
2	13,716	12,213	112	11,127	1,799	12,926
3	1,154	2,940	16	3,223	1,036	4,259
4		1,158		920	684	1,604
5		481		348	400	748
6		212		52	192	244
7		103		11	104	115
8		66		9	60	69
9		29		2	46	48
10		26		2	27	29
11		18		2	12	14
12		10			10	10
13		2			7	7
14		3			1	1
15		1			2	2
16		1				
19		1			1	1
29		1			1	1
All	133,560	133,560	4,452	123,480	10,080	133,560

Table A.5. Frequencies of deviations between original counts and noisy counts after random noise only, *vs.* after CTA adjustment, and frequencies of deviations between perturbed counts after random noise and after different CTA instances.

	Random noise (non-additive)	**CTA (additive)**
Variance of Deviations	0.4691	0.8543
Information Loss	0.57	0.59

Table A.6. Variance of the deviations between original counts and noisy counts, and Information Loss (based on Hellinger's Distance) due to perturbation before and after CTA. Even though the variance increases substantively due to the further deviations caused by CTA, the information loss measure (ranged between 0 and 100 where 0 = "no information loss") reports only low additional loss of information due to the adjustment (i.e. additional information loss is 0.02).

Appendix A.4 – Complex Case

Lower a priori bound for non-sensitive bottom level cells	**CTA Weights**	**(a) Small instance**		**(b) Big instance**	
		Ones	**Twos**	**Ones**	**Twos**
$\min(D; y_i)$	$y_i^{-\gamma}$	0	25	6	389
$\min(D; y_i)$	$b^{\lambda(y_i)} y_i^{-\gamma}$	0	24	6	389
$\min\big(\mathrm{D}; \max(y_i - (N+1); 0)\big)$	$y_i^{-\gamma}$	10	2	62	99
$\min\big(\mathrm{D}; \max(y_i - (N+1); 0)\big)$	$b^{\lambda(y_i)} y_i^{-\gamma}$	4	0	61	106

Table A.7. Frequency table of sensitive cells (i.e. ones and twos) after the initial CTA execution phase in the complex case with D=10, N=2 and b=2. Instance (a) has a total number of 4,452 cells and (b) 16,800 cells.

y	**Before CTA**	**Phase 1**	**Second Phase – Iteration …**						**Final Phase***
			1	**2**	**3**	**4**	**5**	**5***	
0	1,474	1,488	1,552	1,564	1,570	1,570	1,570	1,075	1,076
1	-	61	16	10	4	5	4	1	0
2	-	106	31	14	6	2	2	1	0
3	1,265	832	872	872	878	877	880	553	554
4	371	558	571	591	590	594	590	381	381
>5	13,690	13,755	13,758	13,749	13,752	13,752	13,754	3,281	3,281
All	16,800	16,800	16,800	16,800	16,800	16,800	16,800	5,292	5,292
CPU time	-	3,335	408	6	6	3	3	-	-

Table A.8. Frequency table of counts up to 5 and larger for the big instance example. After iteration 5 of the second phase there will be no improvement. Since six cells are still sensitive, we need a final phase in which only bottom level cells (denoted by *) will be considered for the hierarchical aggregation. Among the 5,292 bottom level cells there are two sensitive cells that must be rounded to the base of 3 prior to the aggregation. CPU times (in seconds) are displayed in the last row.

|z|	Frequencies of deviations between original cells and cells after …	
	Random noise plus hierarchical aggregation of bottom level cells	**CTA plus hierarchical aggregation of bottom level cells**
0	3,342	4,335
1	4,952	6,135
2	3,307	3,377
3	1851	1,484
4	991	674
5	582	353
6	410	171
7	252	100
8	196	83
9	160	55
10	128	21
>11	629	12
All	16,800	16,800

Table A.9. Frequency table of deviations between original and noisy counts with or without CTA adjustment - each time with hierarchically aggregated bottom level cells (i.e., additive hypercube in both cases). The maximum absolute deviation is 175 (without CTA) *vs.* 16 (with CTA).