An open-source tool for experimenting with noise-based perturbation schemes

F. Ricciato, M. Stocchi, F. Bach, A. Bujnowska and W. Kloek

Keywords: *Statistical Disclosure Control; Differential Privacy; Utility vs. privacy risk.*

1 Introduction

The general approach to Statistical Disclosure Control (SDC) based on random perturbation of the published data can be implemented in various ways, and different techniques currently are being considered for application to official population statistics and censuses [1, 2, 3, 4]. While it is commonly accepted that such methods in general trade-off *utility* versus privacy *risk*, there is no consensus yet in the community of SDC experts as to how risk and utility should be defined, measured and ultimately quantified. Our focus here is exclusively on the utility aspect. We present an open-source tool for comparing experimentally the impact of different perturbation methods on published data, and in this way facilitate further research by statistical offices. In this contribution we outline the general workflow of the proposed tool, motivating the main design choice and indicating directions for future extensions.

2 Overview and motivations

The core workflow is depicted in Fig. 1. Let us now focus on the plain non-perturbed branch at the top (in blue color). The micro-data set $\boldsymbol{z}$ consists of N records (rows) of K variables (columns) of interest for the following tabulation. We assume all variables are discrete (possibly categorical). For the generic variable k ($k = 1, \ldots, K$) let $a_k \in \{2, 3, \ldots\}$ denote the integer number of possible values in the micro-data set, i.e., the alphabet size. The number of possible record configurations is therefore $A \stackrel{\text{def}}{=} \prod_{k=1}^{K} a_k$. Through counting, the micro-data set $\boldsymbol{z}$ is transformed into the $K-$dimensional table (hypercube) $\boldsymbol{x}^\star$, with label '$\star$' in apex denoting "non-perturbed" aggregation. In general, each variable may be re-encoded into a (smaller) alphabet of size $b_k \in \{2, 3, \ldots, a_k\}$ by value aggregation (generalisation) so that the total number of hypercube cells is $B \stackrel{\text{def}}{=} \prod_{k=1}^{K} b_k \leq A$. It is clear that in absence of some value aggregation, i.e. when $a_k = b_k$ for every dimension k, then $B = A$ and the transformation $\boldsymbol{z} \rightarrow \boldsymbol{x}^\star$ is directly invertible: in this case, there is no information loss and the hypercube is just a different representation of the original micro-data set. Conversely, when value aggregation is applied, we may qualitatively expect that larger gaps $A - B$ implies larger (potential) information loss, increasing the difficulty of a full reconstruction of the original data set $\boldsymbol{z}$.
A generic statistical function $y = f(\boldsymbol{x})$ is applied on the tabular data to obtain the final desired value y (considered scalar in this contribution for the sake of simplicity). In the simplest case, y may just return the value of a particular cell (whose coordinates are given as input parameter to $f(\cdot)$). In a more elaborate scenario, y may represent one of the regression coefficients of a linear model fitted to the data, and in general the result of some statistical method represented by function $f(\cdot)$. The other two branches in Fig. 1 (in red color) represent two distinct perturbation schemes (of course, one may add further branches to test an arbitrary number of different schemes). Perturbation is introduced to counteract database reconstruction attacks [5, 6] and, like value aggregation (generalisation), it represents a source of information loss.

In the current version of our tool three different perturbation schemes are implemented. Let apex j denote a particular perturbation scheme. Whether noise is added before ("pre-tabular") or after tabulation, the resulting output is a table (hypercube) $\boldsymbol{x}^{(j)}$. The application of function $f(\cdot)$ to this table produces the perturbed final value $y^{(j)}$. In general, for any non-trivial perturbation scheme j, the perturbed value will differ from the non-perturbed one, i.e. $y^\star \neq y^{(j)}$. However, what matters in terms of utility is *how much* it differs, and whether the difference is *significant* or not. But in which terms, according to which criterion the difference between $y^\star$ and $y^{(j)}$ should be judged to be "significant"?

In a recent paper [7] the authors have conducted a comparison of perturbed and non-perturbed values based on real-world data (from US census) and found that for hypercube cells with small values, as encountered for racial/ethnic minorities in small areas, the *relative difference* $r \stackrel{\text{def}}{=} \frac{y^{(j)}-y^\star}{y^\star}$ could be very large. Therein the authors take the relative difference r as the yardstick to judge whether the perturbation error is *significant* or not. In this short contribution we propose to consider an alternative perspective aimed at comparing value ranges (intervals) rather than point values.

We start from the obvious consideration that the micro-data set $\boldsymbol{z}$ is itself the product of a measurement process and, as such, it is unavoidably affected by (different kinds of) errors and uncertainty occurring during the measurement process, i.e., by "measurement noise". In other words, even the non-perturbed versions $\boldsymbol{x}^\star$ and $y^\star$ cannot be considered noise-free, and should be rather seen as a particular realisation of a (noisy) measurement process. That means the perturbation scheme j should not be seen as injecting noise into an otherwise noise-free data set, but rather as adding an *extrinsic* noise component (random perturbation designed for SDC) on top of the *intrinsic* measurement noise component that is already embedded in the original data before perturbation. In this perspective, the amount of intrinsic noise represents the natural reference for assessing whether the amount of additional extrinsic noise is *significant* or not. Of course, quantifying the amount of intrinsic noise from a single realisation $\boldsymbol{z}$ of real-world data is a challenging task, and any such evaluation should be based on accurate knowledge (and modelling) of the measurement process itself. While a formal analysis can be conducted analytically in the most simple scenarios (e.g. y representing single cell counts), for more sophisticated scenarios we must resort to numerical simulations. To this aim, we have conceived our tool structured according to the scheme in Fig. 2.

In the current version the measurement process, denoted by $g(\cdot)$ in Fig. 2, is modelled as a random sampling process where each record is picked with probability ξ independently from other records. Starting from a source micro-data set that represents the ideal "ground truth" population (error-free) we build M independent micro-data sets by randomly sampling with rate ξ ($\xi = 0.9$ in our experiments). In this way, the generic m^{th} data set $\boldsymbol{z}_m$ ($m = 1, \ldots, M$) represents a particular realisation of a census process where population units are missed with probability $1 - \xi = 0.1$. In the current version, sampling error represents the only source of intrinsic noise, but future versions of the tool could implement more elaborate data generation models $g(\cdot)$ aimed at capturing other sources of measurement error [8], possibly integrating synthetic generation tools developed by other researchers.

For each micro-data set, we compute the non-perturbed table $\boldsymbol{x}^\star_m$ plus three perturbed versions $\boldsymbol{x}^{(j)}_m$, $j = 1, \ldots, 3$ obtained with the different perturbation listed below. After applying the statistical function of choice, for each perturbation method j we obtain a vector of M values $\boldsymbol{y}^{(j)} \stackrel{\text{def}}{=} y^{(j)}_1, \ldots, y^{(j)}_M$ and from there compute the Empirical Cumulative Distribution Function (ECDF) that will be denoted by $C^{(j)}$. Summary descriptive indicators (e.g., centrality and dispersion measures) can be computed to compare the range of final values with and without perturbation.

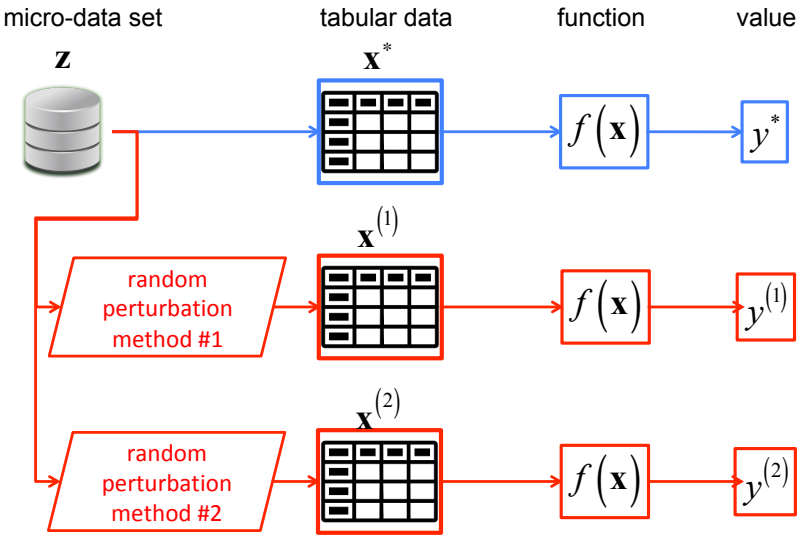

Figure 1: From micro-data through tabulated hypercube to final statistics. Each parallel branch implements a different perturbation scheme.

3 Illustrative numerical results

Our tool is implemented in C++ and R. The rationale behind this choice is to achieve a good trade-off between efficiency and interoperability. The formation of hypercubes was implemented in C++ in order to minimize execution time. Visualisation and scientific operations are implemented in R. In order to maximize interoperability, we link our project to the `LibR` software library to ease data exchange between the two programming languages. This enables the reuse of any available R package, including for instance `ptable` [9]. The current version of the tool implements the following perturbation schemes, all with configurable parameters:

1) The Cell-Key (CK) method as originally described in [1, 10, 11] with p-tables built using the R package `ptable` [9].

2) The Differential Privacy (DP) scheme, introduced in [12, 13], with unbounded noise drawn from the (continuous) Laplace distribution.

3) The DP scheme with unbounded noise drawn from two-sided (discrete) geometric distribution [14, 15].

In our preliminary experiments we set the parameters of all methods to match the same common value for the variance of the (extrinsic) perturbation noise ($V = 8$).
We tested our tool on a synthetic micro-data set mimicking the 2021 EU census hypercube 9.2 (Reg. (EU) 2017/712), with $N = 820{,}000$ micro-data records with $K = 4$ total variables (geographical information at NUTS 3 level, sex, age in 5-year bands, year of arrival in the country). Taking such root data set as the "ground truth" error-free population, the tool samples randomly (at rate $\xi = 0.9$) and generates $M = 100$ different hypercubes for each perturbation method. In this way, for each hypercube cell, we obtain $M \times J$ total values, where $J = 4$ indicates the number of tested schemes (three perturbation schemes plus the non-perturbed case). For this initial test, the considered statistical function $f(\cdot)$ is a simple cell selection.
We performed initial experiments executing the code on a single server machine Intel Xeon CPU E5-2620 v2 (2.10 GHz). The total computation time was about 2 seconds for each hypercube, while the cell selection and ECDF computation caused no significant additional overhead. In Fig. 3 we plot the ECDF (on $M = 100$ trials) obtained

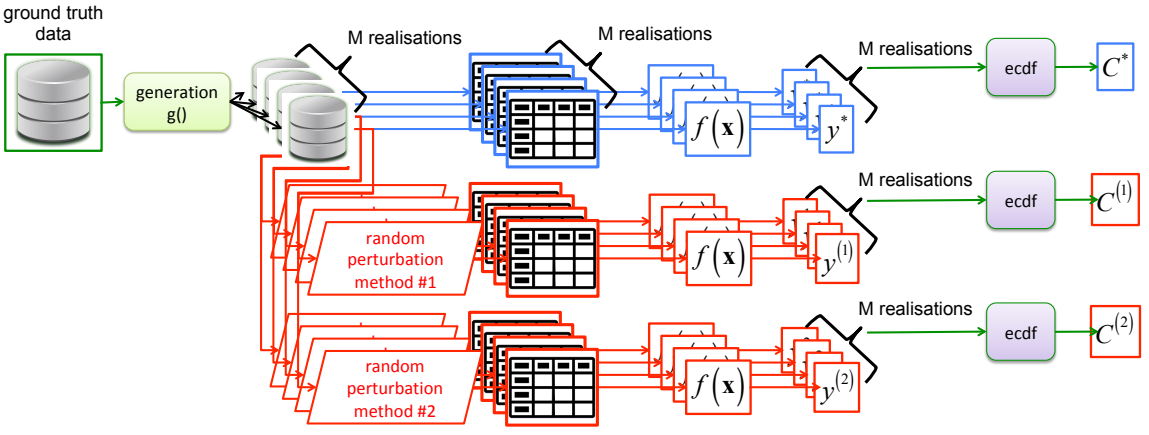

Figure 2: Multiple micro-data set and final statistics are obtained through a stochastic process $g(\cdot)$ representing the measurement process. The empirical distributions of final values are compared across different perturbation schemes.

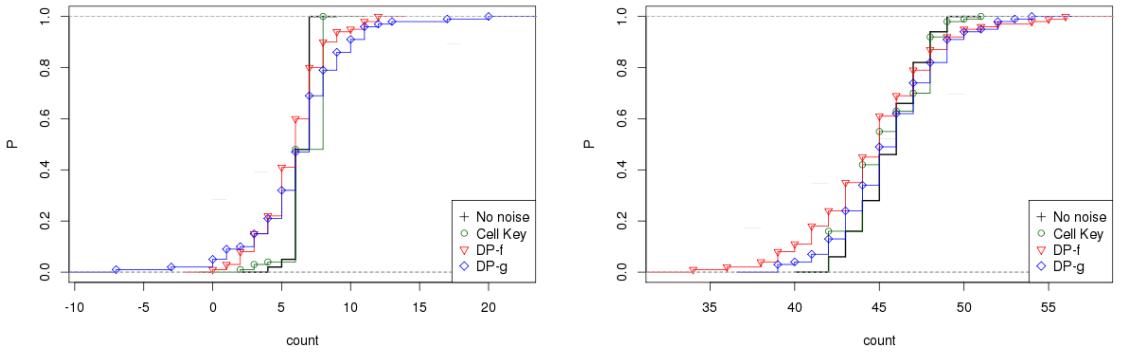

Figure 3: ECDF for two arbitrary cells with smaller (left) and larger (right) values.

with the different perturbation schemes. As expected, all perturbation schemes produce ECDF curves that are somewhat more spread than the reference non-perturbed one, and of course the difference between perturbed and non-perturbed curves is more noticeable for low-value cells (left plot). However, relative to the *intrinsic* noise variance (before perturbation), the additional contribution of *extrinsic* noise variance (after perturbation) seems to be rather contained in this specific scenario. Also, the CK method appears to exhibit systematically smaller variation than DP for all cells.

4 Outlook

At the current stage of the work, the above observations are purely preliminary and do not represent conclusive benchmarking results about the different schemes. They serve only to illustrate in which way our tool can support further empirical investigations towards a better understanding and quantification of the utility loss incurred by the different schemes.

Our tool can be extended in various directions, including more elaborate statistical functions $f(\cdot)$ and data generation models $g(\cdot)$, possibly linking to synthetic population generators developed by other researchers. In agreement with the European Commission strategy [16] our tool is completely open-source: the source code and the synthetic data used for initial tests will be made publicly available on https://github.com/eurostat.

References

[1] Gwenda Thompson, Stephen Broadfoot, and Daniel Elazar. Methodology for the automatic confidentialisation of statistical outputs from remote servers at the Australian Bureau of Statistics. In *Joint UNECE/Eurostat work session on statistical data confidentiality*, 10 2013. URL https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2013/Topic_1_ABS.pdf.

[2] Laszlo Antal, Maël-Luc Buron, Annu Cabrera, Tobias Enderle, Sarah Giessing, Junoš Lukan, Eric Schulte Nordholt, and Andreja Smukavec. Harmonised protection of census data. https://ec.europa.eu/eurostat/cros/content/harmonised-protection-census-data_en, 2017. Accessed on 26 Aug 2020.

[3] John M. Abowd. The U.S. Census Bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2867, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. DOI: 10.1145/3219819.3226070. URL https://doi.org/10.1145/3219819.3226070.

[4] S. Ruggles, C. Fitch, D. Magnuson, and J. Schroeder. Differential privacy and census data: implications for social and economic research. *AEA Papers and Proceedings*, 109:403–408, 2019. DOI: 10.1257/pandp.20191107.

[5] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210, 01 2003. DOI: 10.1145/773153.773173. URL http://www.cse.psu.edu/~ads22/privacy598/papers/dn03.pdf.

[6] Simson L. Garfinkel, John M. Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data. *Queue*, 16(5):28–53, October 2018. ISSN 1542-7730. DOI: 10.1145/3291276.3295691. URL https://doi.org/10.1145/3291276.3295691.

[7] A. Santos-Lozada, J. Howard, and A. Verdery. How differential privacy will affect our understanding of health disparities in the United States. *PNAS*, 2020. DOI: 10.1073/pnas.2003714117.

[8] J. Billiet and H. Matsuo. Non-response and measurement error. In L. Gideon, editor, *Handbook of Survey Methodology for the Social Sciences*, pages 149–178. Springer, New York, 2012. DOI: 10.1007/978-1-4614-3876-2_10.

[9] ptable. https://github.com/sdcTools/ptable, 2020.

[10] Bruce Fraser and Janice Wooton. A proposed method for confidentialising tabular output to protect against differencing. In *Monographs of Official Statistics: Work Session on Statistical Data Confidentiality*, pages 299–302, 11 2005. URL https://op.europa.eu/en/publication-detail/-/publication/bfea1d7d-bc66-4590-933d-90e6f33738c1.

[11] N. Shlomo and C. Young. Invariant post-tabular protection of census frequency counts. In J. Domingo-Ferrer and Y. Saygın, editors, *Privacy in Statistical Databases*, volume 5262 of *Lecture Notes in Computer Science*, pages 77–89. Springer, Berlin, Heidelberg, 2008. DOI: 10.1007/978-3-540-87471-3_7.

[12] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014. DOI: 10.1561/0400000042.

[13] H. Page, C. Cabot, and K. Nissim. Differential privacy: an introduction for statistical agencies. *NSQR, Government Statistical Service*, 2018.

[14] S. Petti and A. Flaxman. Differential privacy in the 2020 US census: what will it do? Quantifying the accuracy/privacy tradeoff. *Gates Open Research*, 3, 2019. DOI: 10.12688/gatesopenres.13089.2.

[15] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6): 1673–1693, 2012.

[16] Open Source Software strategy 2020-2023 – Think Open. Communication to the Commission COM(2020) 7149, 2020. URL https://ec.europa.eu/info/sites/info/files/en_ec_open_source_strategy_2020-2023.pdf.