TRANSACTIONS ON DATA PRIVACY 17 (2024) 147–179

Statistical disclosure control for numeric microdata via sequential joint probability preserving data shuffling

Elias Chaibub Neto[*]

[*]2901 Third Avenue, Suite 330, Seattle, 98121, USA.

E-mail: `elias.chaibub.neto@sagebase.org`

Received 14 November 2023; received in revised form 18 July 2024; accepted 2 August 2024

Abstract. Traditional perturbative statistical disclosure control (SDC) approaches such as microaggregation, noise addition, rank swapping, etc, perturb the data in an "ad-hoc" way in the sense that while they manage to preserve some particular aspects of the data, they end up modifying others. Synthetic data approaches based on the fully conditional specification data synthesis paradigm, on the other hand, aim to generate new datasets that follow the same joint probability distribution as the original data. These synthetic data approaches, however, rely either on parametric statistical models, or non-parametric machine learning models, which need to fit well the original data in order to generate credible and useful synthetic data. Another important drawback is that they tend to perform better when the variables are synthesized in the correct causal order (i.e., in the same order as the true data generating process), which is often unknown in practice. To circumvent these issues, we propose a fully non-parametric and model free perturbative SDC approach that approximates the joint distribution of the original data via sequential applications of restricted permutations to the numerical microdata (where the restricted permutations are guided by the joint distribution of a discretized version of the data). Empirical comparisons against popular SDC approaches, using both real and simulated datasets, suggest that the proposed approach is competitive in terms of the trade-off between confidentiality and data utility.

Keywords. Statistical disclosure control; perturbative methods; restricted permutations

1 Introduction

In this paper we address the problem of statistical disclosure control (SDC) for data publishing in the particular context of numerical microdata files. (That is, where the data is organized into a file with the rows usually containing the data records and columns containing the attributes/variables measured for each data record.) Such datasets are widely used in most fields of science, and the current trend towards open and transparent science means that there is an ever increasing need for publishing datasets supporting scientific research. Often times, the research involves privacy sensitive information, and these supporting datasets need to be masked (perturbed) in order to reduce disclosure risk before they can be shared publicly. Quite importantly, the amount of perturbation should ideally be enough to reduce the disclosure risk to an acceptable level, while retaining as much scientific utility as possible. (Business operations represent another case in point for the need of SDC for data publishing of microdata.)

While there is a rich literature on SDC methodology for numeric microdata [25, 13], most of the perturbation approaches are "ad-hoc" in the sense that while they manage to preserve some particular aspects of the data, they end up modifying others. (E.g., while application of zero mean additive Gaussian noise to a variable approximately preserves its mean, it also increases its variance.) Even more principled perturbative approaches, such as data shuffling [18], are unable to preserve non-monotonic associations in the data.

Fully conditional specification (FCS) synthetic data approaches, on the other hand, aim to generate new datasets that follow the same joint probability distribution as the original data [11]. These approaches, however, rely on parametric statistical models, whose parameters need to be estimated from the data, or rely on non-parametric machine learning models, which need to fit well the data in order to generate credible and useful synthetic data. Another important drawback of these approaches is that they tend to perform better when the variables are synthesized in the correct causal order (i.e., in the same order as the true data generating process), which is often unknown in practice.

In this paper, we propose a fully non-parametric and model free data perturbation approach that circumvents important drawbacks of both perturbative and synthetic data based SDC methods. The approach, denoted sequential joint probability preserving data shuffling (SJPPDS), is based on sequential applications of restricted permutations to the numerical microdata, which are guided by the joint probability distribution of a discretized version of the data. As such, the approach is able to approximate the joint probability distribution of the original data without relying on parametric or non-parametric models, and without requiring any domain knowledge about the true data generating process. In practice, this means that the approach can be effective even when applied to small and noisy datasets, which tend to be challenging for synthetic data approaches that depend on models for fitting the data. Furthermore, in addition to approximately preserving the joint association structure of the data, the SJPPDS approach, by construction, produces a masked/perturbed dataset with the exact same marginal distributions as the original data. We implement two versions (full and simplified) of the SJPPDS method (described in Algorithms 1, 2, and 3 of Section 3).

Following [8], we evaluate the tradeoff between data confidentiality and utility using a combination of multiple disclosure risk (DR) and information loss (IL) metrics. (Adoption of multiple metrics is generally advisable, since different metrics capture different aspects of the similarity between the original and masked datasets.) We compare the proposed approach against popular SDC approaches, using two real business microdata datasets, as well as, 60 simulated datasets. Our experimental results favor the SJPPDS approaches in terms of the trade-off of general data utility and disclosure risk.

2 Related work

Statistical disclosure control (SDC) is a mature field with many well established methods for the masking of numerical microdata [25, 11, 13]. Traditional perturbative methods include: (i) microaggregation [5, 23], where the original data records are first combined into small aggregate groups before their values are replaced by the mean of the aggregate groups; (ii) noise addition [2], where masking is obtained by adding (independent or correlated) noise to the original data; and (iii) rank-swapping [17], where each variable is first ranked in ascending order before each ranked value is swapped with another ranked value randomly chosen within a restricted range (such that the ranks of two swapped values cannot differ by more than a fixed percentage of the total number of records).

Synthetic data approaches have also been proposed for SDC, where instead of directly changing the original data values, completely new values are sampled from appropriate probability distributions that, hopefully, capture the essential statistical properties of the original data. For instance, under the assumption of multivariate normality, information preserving statistical obfuscation (IPSO) methods [1, 4, 14] aim to generate synthetic data that preserves the outputs of regression models (e.g., regression coefficients and/or covariance matrices).

Conceptually, the analytical properties of the original data are fully captured by the joint probability distribution (j.p.d.) of all variables in the dataset. The perturbative and synthetic data SDC approaches listed above, however, are not able to preserve the j.p.d. of the data. Fully conditional specification (FCS) methods [11], on the other hand, aim exactly at simulating synthetic data with the same j.p.d. as the original data. The idea is to fully factorize the j.p.d. of the data, $P(X_1, X_2, X_3, \ldots, X_p)$, into a series of conditional distributions, $P(X_1 \mid X_2, \ldots, X_p)P(X_2 \mid X_3, \ldots, X_p) \ldots P(X_p)$, and then sequentially model and simulate one variable at a time, conditionally on the previous ones. The quality of the synthetic data generated using FCS methods depends, however, on the quality of the original data (as the conditional distribution models need to be estimated from the original data). While parametric models can be used to estimate the conditional distributions, these models need to be chosen very carefully, and the FCS implementation based on the classification and regression tree (cart) model [21] is often seen as the go to approach for FCS data synthesis, as it can flexibly model unusual data distributions and capture non-linear associations without requiring specification of the conditional distributions, and has been shown to provide the best empirical results in practice [12, 19]. Despite these advantages, the quality of cart based data synthesis is still influenced by the order of the variables. While there is no standard way to select the variable order, the general recommendation is to try to emulate the causal ordering behind the data generating process giving rise to the original data. The problem, however, is that, in many applications this knowledge is not available.

Finally, a related approach to the shuffling methods proposed in this paper is the data shuffling procedure proposed by Muralidhar and Sarathy [18], which we denote here as d-shuffle. This procedure is based on the conditional distribution approach to SDC, and the basic idea is to generate synthetic data values from the conditional distribution of the confidential variables given the non-confidential variables and then perform reverse-mapping of these synthetic values to the values of the original confidential variables. (Where the reverse mapping is obtained by replacing the ordered values of the synthetic variables by the ordered values of the original confidential variables). In practice, the synthetic data values are sampled from a simplified copula-based approach which only requires the estimation of the rank-order correlation matrix of the original (confidential and non-confidential) variables and the ranks of the non-confidential variables. Importantly, in situations where all variables in the data are confidential, the approach can still be applied by simply generating an independent multivariate normal data with mean zero and covariance matrix set to the estimated rank-order correlation matrix and then reverse-map this synthetic data set to the original data. Similarly, to the SJPPDS methods proposed in this paper, the d-shuffle approach also preserves the marginal distribution of the data and tends to achieve low disclosure risk and high data utility. One important caveat of the d-shuffle procedure is that it is unable to preserve complex non-monotonic associations in the data, as such associations cannot be well captured by rank-order correlations. (Figure A1 in the Appendix provides an illustrative example of this issue.) Furthermore, noisier datasets, for which it can be difficult to obtain a good estimate of the rank-order correlation matrix can also be challenging for the d-shuffle approach.

3 The proposed approach

Before we describe the SJPPDS approach we first illustrate (using a toy example) how restricted permutations can be used as a data perturbation mechanism that (approximately) preserves the associations of numerical data. Consider the data from two highly correlated numerical variables, X_1 and X_2, in rows 1 and 2 of Table 1, simulated from a bivariate normal distribution. Here we describe how to perform restricted permutations of the X_1 data relative to the X_2 variable.

Table 1: Restricted permutation example

X_1	8.87	9.57	9.61	9.36	9.75	10.51	10.01	9.67	10.29	11.42	12.11	11.64
X_2	9.66	10.09	10.52	10.54	10.80	11.19	11.24	11.47	11.61	11.96	12.23	12.39
C_2	1	1	1	1	2	2	2	2	3	3	3	3
$X_1^\star$	9.61	9.36	8.87	9.57	9.75	10.51	9.67	10.01	11.64	12.11	10.29	11.42

The first step is to obtain a categorical version of the X_2 variable, denoted C_2, by discretizing the X_2 data into n_c categories. In the example in Table 1 we discretize X_2 into $n_c = 3$ categories/levels, denoted as, "1", "2", and "3" as shown in the C_2 row. (The discretization is performed by splitting the range of the X_2 data into 3 equally sized bins, and assigning the categorical labels "1", "2", and "3" to the X_2 values that fall in each of these bins.) A restricted permutation of the X_1 data is then obtained by separately shuffling the values of the X_1 data within each level of the C_2 variable, as shown in the fourth row of Table 1, where the shuffled X_1 values are denoted by $X_1^\star$. Figure 1 plots the values X_1 against X_2 for the original data (panel a) and for the restricted permutation data (panel b). Clearly, the restricted permutation approach preserves well the association between the X_1 and X_2 variables. Observe, as well, that the marginal distribution of the shuffled data $X_1^\star$ is, by construction, identical to the marginal distribution of the original data X_1.

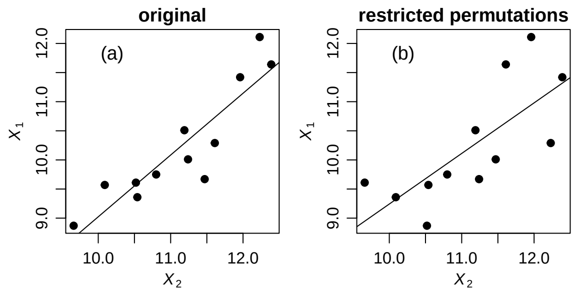

Figure 1: Restricted permutation example.

An important tuning parameter of the restricted permutation approach is the number of categories/levels, n_c, used in the discretization of X_2. The larger the n_c, the better is the association preservation, as illustrated in Figure 2, where we now simulate $n = 10,000$ records from the same bivariate normal distribution as in Table 1 ($\boldsymbol{\mu} = (10, 11)^T$ and $\sigma_{11} = 1$, $\sigma_{22} = 1$, and $\sigma_{12} = 0.9$).

On the other hand, the adoption of large values for the n_c parameters will also lead to increased disclosure risk, as the larger the n_c, the smaller the overall amount of data shuffling (and in the extreme case when n_c equals the number of records in the data, there is no shuffling of the data).

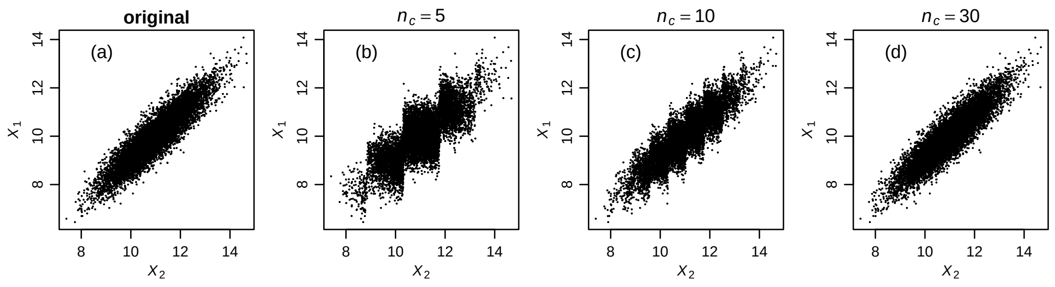

Figure 2: Effect of the number of categories/levels parameter, n_c, on the preservation of the association between X_1 and X_2 obtained by the restricted permutation approach.

3.1 Sequential joint probability preserving data shuffling

We now describe how to perform (approximate) joint probability preserving data shuffling. The basic idea is to create a categorical version of the numeric microdata and then perform restricted permutations of the numeric data guided by the fully factorized joint probability distribution of the categorical data,

$$P(C_1, C_2, \ldots, C_p) = P(C_1 \mid C_2, \ldots, C_p)\, P(C_2 \mid C_3, \ldots, C_p)\, \ldots\, P(C_{p-1} \mid C_p)\, P(C_p)\,. \quad (1)$$

Algorithm 1 describes the approach in detail. (Note that it can be seen as a fully non-parametric, model free, and perturbative analog of the fully conditional specification data synthesis paradigm.)

To fix ideas, suppose that the numerical microdata, $\boldsymbol{X}$, contains 4 attributes, X_1, X_2, X_3, and X_4, each of which is discretized into $n_c = 10$ classes (named as "1", "2", ..., "10") in order to produce the associated categorical variables C_1, C_2, C_3, and C_4. Suppose, as well, that the microdata contains 10,000 records, so that both the numeric microdata and its categorical version, $\boldsymbol{C}$, have dimension $n = 10{,}000$ by $p = 4$. Now, consider the full factorization of the joint probability distribution of the categorical data,

$$P(C_1, C_2, C_3, C_4) = P(C_1 \mid C_2, C_3, C_4)\, P(C_2 \mid C_3, C_4)\, P(C_3 \mid C_4)\, P(C_4)\,. \quad (2)$$

Application of Algorithm 1 to the numeric microdata goes as follows. The outer *for-loop* (starting at line 4) captures the separate terms of equation (2). For $i = 1$, the algorithm deals with the $P(C_1 \mid C_2, C_3, C_4)$ term. Line 5 creates the $\boldsymbol{C}^*$ matrix by selecting columns 2, 3, and 4 of $\boldsymbol{C}$. Line 6 creates the vector of combinations of the categorical variables in $\boldsymbol{C}^*$ ($lcombs$) by pasting together the levels of the C_2, C_3, and C_4 variables into a character string. (For example, if the values of C_2, C_3, and C_4 for the first record in the data are given, respectively, by "9", "2", and "6", then the value at the first position of $lcombs$ is given by the string "9.2.6". Note that for $n_c = 10$ there are at most $10 \times 10 \times 10 = 1{,}000$ possible distinct combinations of levels.) Line 7 obtains the unique combinations/strings observed in the data ($ucombs$), and line 8 counts the number of unique combinations (n_u) in the data. The inner *for-loop* of Algorithm 1 (starting at line 9) effectively performs a restricted permutation of X_1 relative to the combination of the C_2, C_3, and C_4 variables and generates the shuffled numeric data as follows. For each one of the unique combinations in $ucombs$, line 10 finds the indexes of the records that share that combination, and lines 11 and 12 shuffle the numerical microdata values of variable X_1 within these indexes. After running through all unique combinations the result is a dataset where the values of X_1 have been

Algorithm 1: Joint probability preserving data shuffling - full version (JPPDS-f)

```
Data: Microdata, $\boldsymbol{X}$; categorical version of the microdata, $\boldsymbol{C}$
1  Find the number of attributes, $p$, of $\boldsymbol{X}$, i.e., $p \leftarrow$ NumberOfColumns($\boldsymbol{X}$)
2  Find the number of records, $n$, of $\boldsymbol{X}$, i.e., $n \leftarrow$ NumberOfRows($\boldsymbol{X}$)
3  Initialize the matrix with shuffled data with the original microdata, i.e., $\boldsymbol{X}_s \leftarrow \boldsymbol{X}$
4  for $i$ in 1, ..., p-1 do
5      Create a new matrix, $\boldsymbol{C}^*$, by selecting columns of $\boldsymbol{C}$ ranging from $i+1$ until $p$, i.e., $\boldsymbol{C}^* \leftarrow \boldsymbol{C}[, (i+1) : p]$
6      Create a vector of length $n$ containing the character strings obtained by pasting together the columns of $\boldsymbol{C}^*$. This vector contains the combinations of the categorical variables in $\boldsymbol{C}^*$, and is denoted lcombs. Note that some of the combinations in this vector can be repeats
7      Get the vector containing only the unique combinations of the lcombs vector, i.e., $ucombs \leftarrow$ Unique($lcombs$)
8      Get the length of the $ucombs$ vector, i.e., $n_u \leftarrow$ Length($ucombs$)
9      for $j$ in 1, ..., $n_u$ do
10         Find the indexes of the rows of $\boldsymbol{C}^*$ that have the variable combination in $ucombs[j]$, i.e., $idx \leftarrow$ Which($lcombs == ucombs[j]$)
11         Obtain a randomly shuffled version of the indexes in $idx$, i.e., $idx_s \leftarrow$ Shuffle($idx$)
12         For columns ranging from 1 to $i$, replace the data in the rows of $\boldsymbol{X}_s$ indexed by $idx$ by the data in rows $idx_s$ of $\boldsymbol{X}$, i.e., $\boldsymbol{X}_s[idx, 1:i] \leftarrow \boldsymbol{X}[idx_s, 1:i]$
13 Randomly shuffle the rows of $\boldsymbol{X}_s$, i.e., $\boldsymbol{X}_s \leftarrow \boldsymbol{X}_s$[Shuffle($1:n$), ].
Result: Return the shuffled dataset $\boldsymbol{X}_s$
```

shuffled in a way that preserves (approximately) their association with the $\{X_2, X_3, X_4\}$ variables.

For $i = 2$, the algorithm deals with the $P(C_2 \mid C_3, C_4)$ term. Now, line 5 creates the $\boldsymbol{C}^*$ matrix by selecting columns 3 and 4 of $\boldsymbol{C}$. Line 6 pastes together variables C_3 and C_4, and now there are at most $10 \times 10 = 100$ possible combinations in $ucombs$. Quite importantly, note that after the algorithm selects the indexes of the records that will be shuffled (lines 10 and 11), rather than shuffling the X_2 values alone, it shuffles both the X_2 and X_1 values together, as described in line 12. This is done to preserve the association between all four variables, rather than just the association between X_2 and $\{X_3, X_4\}$.

For $i = 3$, the algorithm deals with the $P(C_3 \mid C_4)$ term. Now, $\boldsymbol{C}^*$ corresponds to column 4 alone, and there are only 10 possible combinations in $ucombs$. For each level of the categorical variable C_4, after the algorithm selects the indexes of the records that will be shuffled (lines 10 and 11), it shuffles both the X_3, X_2 and X_1 values together (line 12) to preserve the associations between all four variables.

Finally, note that line 13 of Algorithm 1 performs one last random shuffle of the rows of the matrix $\boldsymbol{X}_s$ (i.e., randomly shuffle the records of $\boldsymbol{X}_s$), because after the completion of steps 1 to 12, the last column of $\boldsymbol{X}_s$ is still identical to the last column of $\boldsymbol{X}$.

It is important to point out that the amount of shuffling performed by Algorithm 1 increases as i increases from 1 to $p-1$ (line 4), simply because the number of possible combinations of the levels of the C variables in the $ucombs$ vector, n_u, decreases as the number of number of columns of the $\boldsymbol{C}^*$ matrix decreases. (In the above toy example, $n_u \leq 1000$

in the first iteration, $n_u \leq 100$ in the second, and $n_u \leq 10$ in the third.) Note that in situations where the number of records is smaller than the number of combinations in the $ucombs$ vector, we have that the number of records sharing the same combination of categorical variables in $ucombs$ will be 1 in most cases, so that Algorithm 1 barely performs any shuffling during its first iterations.

This observation suggests that we might be able to simplify Algorithm 1 to improve its computational efficiency without sacrificing too much its performance. Algorithm 2 provides a simplified version where we remove the outer *for-loop* and only shuffle together the data of the first $p-1$ variables within each level of the last variable. In this case, we are essentially leveraging the following factorization of the joint probability distribution of the categorical variables,

$$P(C_1, C_2, \ldots, C_{p-1}, C_p) = P(C_1, C_2, \ldots, C_{p-1} \mid C_p)\, P(C_p)\,, \tag{3}$$

to guide the restricted permutations of the data.

Algorithm 2: Joint probability preserving data shuffling - simplified version (JPPDS-s)

```
   Data: Microdata, $\boldsymbol{X}$; categorical version of the microdata, $\boldsymbol{C}$
1  Find the number of attributes, $p$, of $\boldsymbol{X}$, i.e., $p \leftarrow$ NumberOfColumns($\boldsymbol{X}$)
2  Find the number of records, $n$, of $\boldsymbol{X}$, i.e., $n \leftarrow$ NumberOfRows($\boldsymbol{X}$)
3  Initialize the matrix with shuffled data with the original microdata, i.e., $\boldsymbol{X}_s \leftarrow \boldsymbol{X}$
4  Get the last column of $\boldsymbol{C}$, i.e., $C_p \leftarrow \boldsymbol{C}[, p]$
5  Get the unique values of $C_p$, i.e., $C_{pu} \leftarrow$ Unique($C_p$)
6  Get the length of the $C_{pu}$ vector, i.e., $n_u \leftarrow$ Length($C_{pu}$)
7  for $j$ in 1, ..., $n_u$ do
8      Find the indexes of $C_p$ that have the value in $C_{pu}[j]$, i.e.,
         $idx \leftarrow$ Which($C_p == C_{pu}[j]$)
9      Obtain a randomly shuffled version of the indexes in $idx$, i.e., $idx_s \leftarrow$ Shuffle($idx$)
10     For the first $p-1$ columns, replace the data in the rows of $\boldsymbol{X}_s$ indexed by $idx$ by
         the data in rows $idx_s$ of $\boldsymbol{X}$, i.e., $\boldsymbol{X}_s[idx, 1:(p-1)] \leftarrow \boldsymbol{X}[idx_s, 1:(p-1)]$
11 Randomly shuffle the rows of $\boldsymbol{X}_s$, i.e., $\boldsymbol{X}_s \leftarrow \boldsymbol{X}_s$[Shuffle($1:n$), ].
   Result: Return the shuffled dataset $\boldsymbol{X}_s$
```

It is important to point out, however, that because the data in the first $p-1$ variables are shuffled together by Algorithm 2, we have that essentially only the data in the last column is shuffled relative to the first $p-1$ columns. Hence, in order to make sure that the data in all columns are shuffled relative to each other we actually implement a sequential version of Algorithm 2 where it is sequentially applied to shifting orderings of the columns of the data, as described in Algorithm 3. (Note that it is also usually beneficial to use the sequential approach described in Algorithm 3 in conjunction with Algorithm 1, since the full JPPDS approach implemented in Algorithm 1 can still fail to adequately shuffle the data in the first columns of the dataset.)

Going back to the toy example with four variables, we have that line 3 of Algorithm 3 shuffles X_4 relative to $\{X_1, X_2, X_3\}$, while the *for-loop* in lines 4 to 7 sequentially apply the JPPDS (full or simplified) to the following data orderings of the data, $\{X_2, X_3, X_4, X_1\}$, $\{X_3, X_4, X_1, X_2\}$, and $\{X_4, X_1, X_2, X_3\}$, so that all variables are shuffled relative to each other. (That is, X_4 is shuffled relative to $\{X_1, X_2, X_3\}$ in line 3; X_1 is shuffled relative to $\{X_2, X_3, X_4\}$ when $i = 1$ in the *for-loop* starting in line 4; X_2 is shuffled relative to $\{X_3, X_4, X_1\}$ for $i = 2$; and X_3 is shuffled relative to $\{X_4, X_1, X_2\}$ for $i = 3$.)

Algorithm 3: Sequential joint probability preserving data shuffling (SJPPDS)

```
Data: Microdata, $\boldsymbol{X}$; number of classes/levels, $n_c$
1 Find the number of attributes, $p$, of $\boldsymbol{X}$, i.e., $p \leftarrow$ NumberOfColumns($\boldsymbol{X}$)
2 Create the categorical version of $\boldsymbol{X}$, denoted $\boldsymbol{C}$, by discretizing each column of $\boldsymbol{X}$
    into $n_c$ categories: $\boldsymbol{C} \leftarrow$ CategorizeData($\boldsymbol{X}, n_c$)
3 Generated the masked data, $\boldsymbol{X}^\star$, by using the JPPDS algorithm:
    $\boldsymbol{X}^\star \leftarrow$ JointProbabilityPreservingDataShuffling($\boldsymbol{X}, \boldsymbol{C}$)
4 for $i$ in 1, ..., p - 1 do
5     Change the order of the columns of $\boldsymbol{X}^\star$, so that the first column is placed last, i.e.,
        $\boldsymbol{X}^\star \leftarrow \boldsymbol{X}^\star[, c(2:p, 1)]$
6     Update the categorical version of the masked microdata, i.e.,
        $\boldsymbol{C}^\star \leftarrow$ CategorizeData($\boldsymbol{X}^\star, n_c$)
7     Update the masked microdata, i.e.,
        $\boldsymbol{X}^\star \leftarrow$ JointProbabilityPreservingDataShuffling($\boldsymbol{X}^\star, \boldsymbol{C}^\star$)
8 Restore column order to the order in the original dataset, i.e., $\boldsymbol{X}^\star \leftarrow \boldsymbol{X}^\star[, c(2:p, 1)]$
Result: Return the masked dataset $\boldsymbol{X}^\star$
```

4 Performance evaluation

In this section we present comparisons of the proposed method against data perturbation approaches, as well as, a synthetic data method, in both real and simulated data experiments. For the real data experiments, we use 2 business microdata datasets, Census and Tarragona, which have been traditionally used to evaluate SDC approaches for numerical microdata. Both datasets are available with the sdcMicro R package [24]. (Note that the Census dataset is named as CASCrefmicrodata in the sdcMicro package.) The Census data is composed of 1080 records on 13 variables (but we only use 12 of the Census dataset variables for our illustrations because one of the variables, PEARNVAL, is a linear combination of the other 12 variables, what causes problems for the calculation of some of the information loss metrics used in our evaluations. In practice, we can always apply the masking methods to the 12 variables and simply compute the masked version of the PEARNVAL variable from the other variables.) The Tarrogana data is composed of 834 records on 13 variables. In addition to being relatively small, both datasets contain variables with very different scales. Furthermore, the variables in the Tarragona dataset show fairly skewed data distributions and contain outliers. For the simulated data illustrations, we performed 2 experiments simulating data from multivariate normal and correlated exponential variables. In each experiment, we simulate 30 distinct $\boldsymbol{X}$ matrices of dimension $n = 500$ by $p = 10$ with different location, scale, and correlation strengths. (See Appendix A1.1 for details.)

4.1 Comparisons against data perturbation approaches

We compared the performance of the proposed method against the following widely used data perturbation approaches in the SDC field: (i) microaggregation, implemented using three distinct grouping methods (namely, the maximum distance to average vector (mdav) method [5], the principal component analysis (pca) method, and the projection pursuit principal components (pppca) method [23]); (ii) noise addition [2] based on independent additive noise and correlated noise; (iii) rank-swapping [17] based on the percentage of ranks to be swapped; and (iv) data shuffling [18] (denoted here as d-shuffle). All perturbation methods were implemented using the sdcMicro R package [24], with the exception of the d-shuffle approach which was implemented as described in Appendix A1.2. (Note that while sdcMicro package provides an implementation of the data shuffling approach, the package's implementation does not handle the case where all variables are confidential.)

In all experiments, we evaluated the tradeoff between data confidentiality and data utility using a combination of 3 disclosure risk and 3 information loss metrics. Disclosure risk (DR) metrics evaluate either re-identification disclosure risk (i.e., the risk that an intruder might be able to determine the subject/entity to whom a given masked data record belongs to) or attribute disclosure risk (e.g., the risk that an intruder can learn about the value of confidential variables). We adopt distance-based record linkage (DBRL) [20, 8] to quantify re-identification disclosure risk, and rank-based interval disclosure (RID) [8] and standard deviation-based interval disclosure (SDID) [15] to quantify attribute disclosure risk. (See Appendix A1.3 for a brief description of these metrics.)

Information loss (IL) metrics can be classified as either general or analysis specific measures [26]. General measures attempt to either directly measure statistical distances between the masked and original datasets, or to measure the closeness of specific distribution parameters (e.g., moments, quantiles, and other summary statistics). Analysis specific measures, on the other hand, compare the results from specific analysis performed in the original and masked datasets (e.g., compare the point estimates and confidence intervals of a regression analysis). Since our goal is to publish masked datasets supporting scientific research or business operations, which might be used by others in unanticipated ways, we adopted general utility metrics in our evaluations. The selected IL measures included the propensity-score based information loss (PS) [26], the probabilistic information loss (PIL) [16], and the covariance bounded information loss metric [10]. Appendix A1.4 provides further details about these selected metrics.

Quite importantly, note that all 3 DR metrics selected for our comparisons (namely, DBRL, RID and SDID) can only, strictly speaking, be applied to masking methods for which there exists a mapping between the original, x_{ij}, and masked values, $x^{\star}_{ij}$. While this mapping is (at first sight) destroyed by the data shuffling process employed by the SJPPDS and d-shuffle approaches, in practice, it is still possible to find a mapping between the original and perturbed data values by sorting the $\boldsymbol{X}$ and $\boldsymbol{X}^{\star}$ datasets according to any arbitrary attribute (column) of the data before computing the DR metrics (analogously to the work of [10]). The precise algorithm is presented in Appendix A1.5 (see Algorithm 4), but the basic idea is to: (i) separately sort the original and masked datasets according to the value of one of their attributes; (ii) compute the selected metric using the sorted datasets; and (iii) repeat steps (i) and (ii) for all attributes, and compute the maximum value across all sortings (i.e., select the most conservative estimate of disclosure risk). Note that this procedure is justified under the permutation paradigm for data anonymization proposed by [7] (and is analogous to the procedure for the computation of the CM3 metric proposed in [10], which is used for the evaluation of disclosure risk in synthetic datasets - see Appendix A1.5 for further details). Observe, as well, that this procedure generates more conservative DR metrics (i.e., larger disclosure risks) for the SJPPDS and d-shuffle methods than the direct calculation of the metrics using the unsorted datasets $\boldsymbol{X}$ and $\boldsymbol{X}^{\star}$ (see Figures A2, A3, and A4 in the Appendix for an illustration). In our evaluations we adopt this "worse case sorted approach" for the evaluation of the SJPPDS and d-shuffle methods, but the standard (unsorted) approach for the evaluation of the microaggregation, noise addition, and rank swapping approaches.

Following reference [8], we combine all these disclosure risk and information loss metrics into a single formula (score) meant to capture the overall confidentiality/utility tradeoff and provide an overall ranking of these SDC methods. Our overall score formula is given by,

$$\text{overall score} = \left[\frac{\text{DBRL}}{6} + \frac{\text{RID}}{6} + \frac{\text{SDID}}{6}\right] + \left[\frac{\text{PS}}{6} + \frac{\text{PIL}}{6} + \frac{\text{CBIL}}{6}\right] , \tag{4}$$

which gives equal weights for the combination of DR metrics (1/6 + 1/6 + 1/6 = 0.5) and combination of IL metrics (1/6 + 1/6 + 1/6 = 0.5). Because the PS metric range is between [0, 0.25] we multiply it by 4, so that it is also bounded between 0 and 1, as all the other metrics used for the computation of the overall score in (4). Note that lower overall scores are preferred since they indicate lower disclosure risks and lower information losses.

For each of the masking methods, we used the above score formula to:

1. First select the best parameter across a grid of 30 tuning/perturbation parameter values (under the constraint that the DBRL metric is below a given fixed threshold).

2. Then to compare the overall performance of the SDC methods based on the selected best parameter score.

Note that we only select tuning parameters that produce DBRL values below a given fixed threshold, because, in practice, we are not interested in masking methods that preserve well data utility at the expense of high disclosure risks. Figure A5 in the Appendix provides an illustrative example of the selection of best tuning/perturbation parameter values conditional on different DBRL thresholds. In our comparisons we adopt a DBRL threshold of 0.2. (We choose this relatively high threshold because the noise-addition methods generated high values of DBRL in the simulated datasets and choosing a lower threshold would lead to discarding these methods from the comparisons. In practice, however, one might be interested in lower threshold values.)

For both the real and simulated data experiments, we adopted the following tuning (perturbation) parameter grids:

- Aggregation parameters in the range $\{2, 3, \ldots, 31\}$ for the microaggregation methods.
- Noise percentage parameter in the range $\{1, 5, 9, 13, \ldots, 117\}$% for the noise addition methods.
- Swap percent parameter in the range $\{2, 4, 6, \ldots, 60\}$% for rank-swapping.
- Number of classes parameter (n_c) in the range $\{10, 20, \ldots, 300\}$ for both the SJPPDS methods.

Note that the d-shuffle approach does not require any tuning.

Figure 3 reports the tradeoffs between information loss and disclosure risk across the grid of tuning/perturbation parameter values for each one of the SDC methods in the Census dataset. In all panels, the blue curve represents the average information loss score,

$$\text{average information loss} = \frac{\text{PS}}{3} + \frac{\text{PIL}}{3} + \frac{\text{CBIL}}{3} , \tag{5}$$

the red curve represents the average disclosure risk score,

$$\text{average disclosure risk} = \frac{\text{DBRL}}{3} + \frac{\text{RID}}{3} + \frac{\text{SDID}}{3} , \tag{6}$$

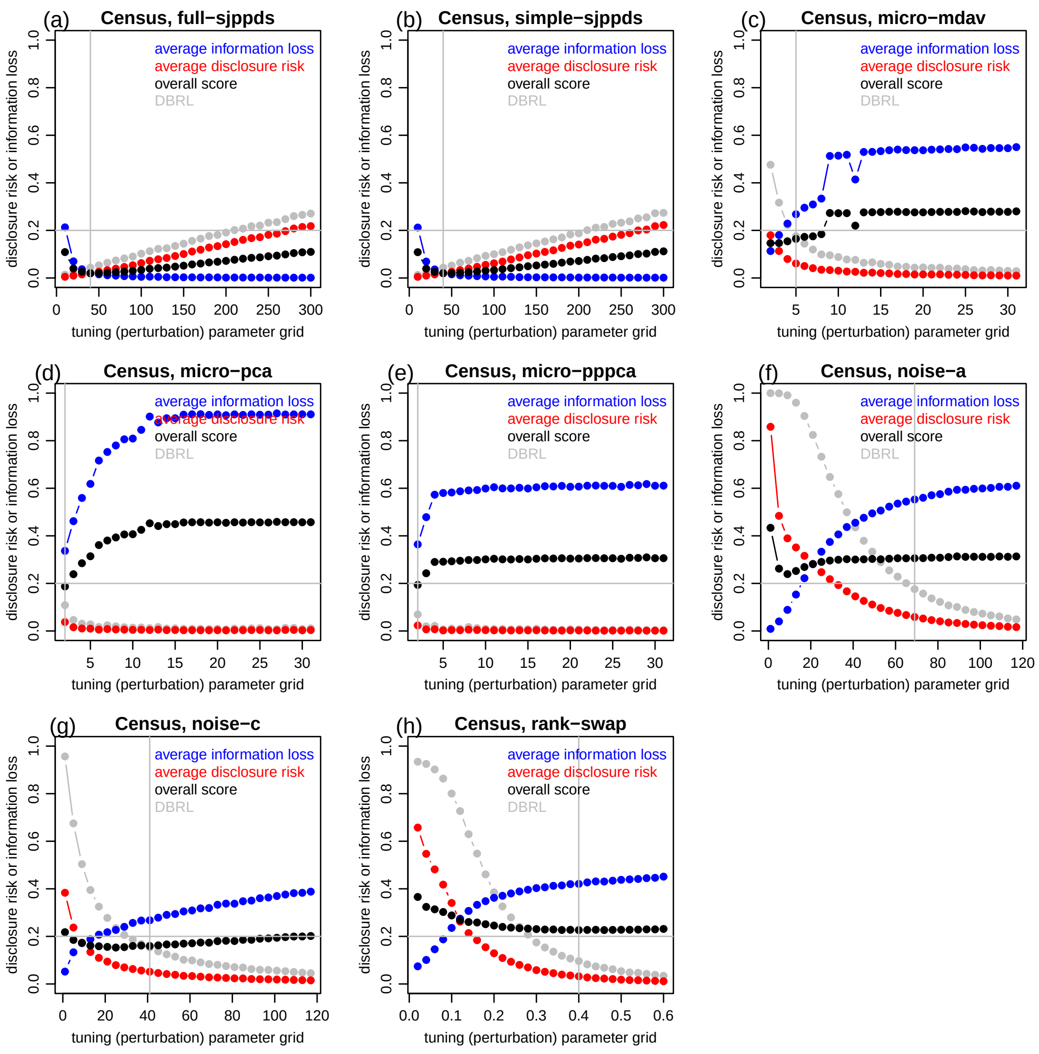

Figure 3: Tradeoff between information loss and disclosure risk in the Census dataset.

the black curve represents the overall score in equation 4 (i.e., the average of the average information loss and average disclosure risk scores), and the grey curve represents the DBRL metric. (Note that each point on each of these curves actually corresponds to the median value from the 30 experiment replications.) The grey horizontal line shows the selected DBRL threshold (i.e., 0.2), while the grey vertical line shows the selected best parameter value (under the constraint that DBRL is less than 0.2). Figure A6 in the Appendix shows the analogous plot for the Tarragona data experiments. (Due to space limitations, we do present the condidentialy/utility tradeoff plots for the simulated datasets, as those would require a separate plot for each one of the 60 simulated datasets.)

Figure 4 reports the results from all experiments, comparing the overall scores of each of the SDC methods. For the real data experiments (panels a and b), the boxplots report the results from 30 replications of the experiments based on the best selected tuning parameter (with the exception of the microaggregation methods, which are deterministic, and therefore based on a single experiment run). For the simulated data experiments (panels c and d) the boxplots report the scores from the 30 distinct simulated datasets. (Each method is first evaluated on a grid of 30 distinct tuning parameter values, but the boxplots only report the results based on the best selected tuning/perturbation parameter.) In terms of the data confidentiality/utility tradeoff measured by the overall score, the SJPPDS approaches outperformed all other methods in all experiments.

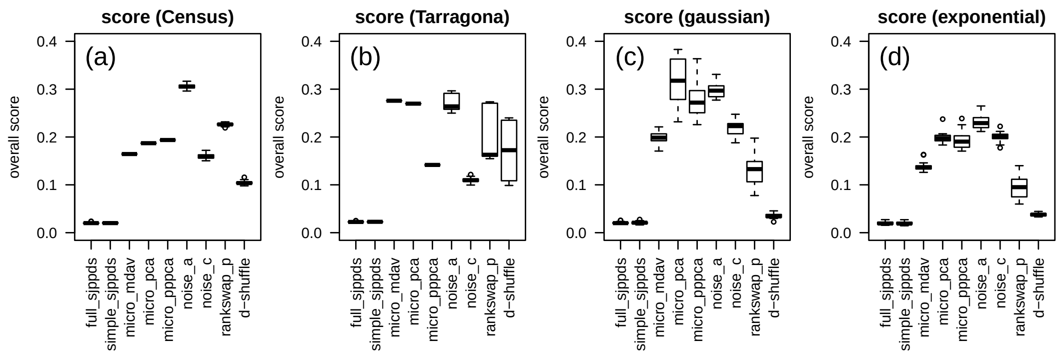

Figure 4: Data perturbation methods results from the real and simulated data experiments.

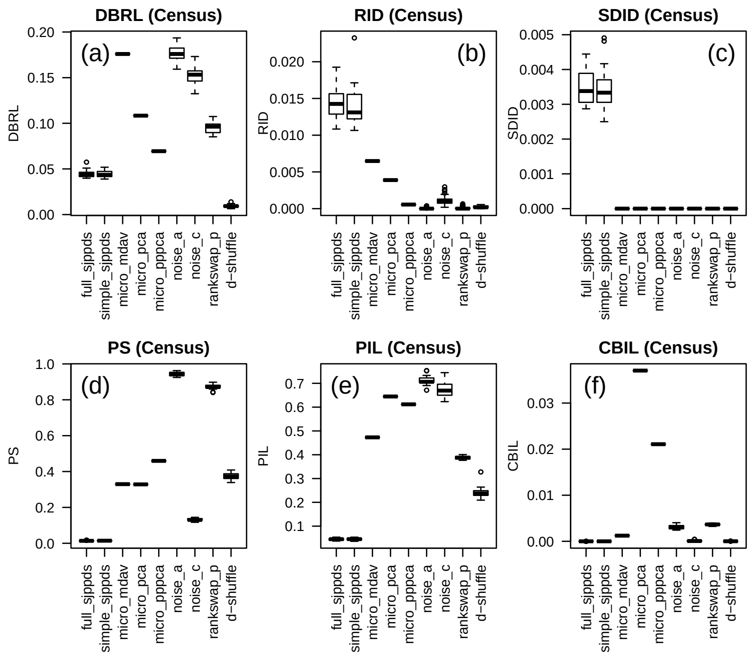

Figure 5: Metrics comparison for data perturbation methods in the Census data experiment.

Figures 5, 6, 7, and 8 present more detailed descriptions of the experimental results looking at each of the DR and IL metrics separately for each one of the experiments. The box-

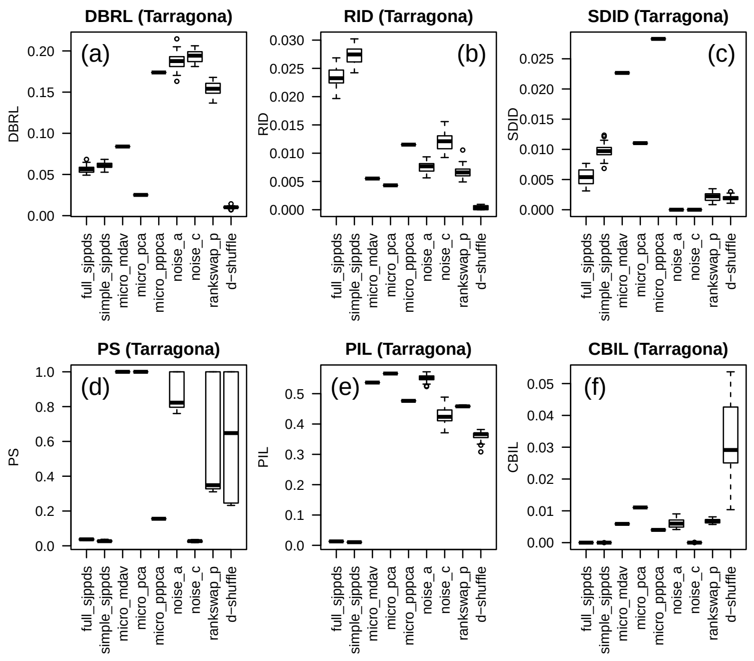

Figure 6: Metrics comparison for data perturbation methods in the Tarragona data experiment.

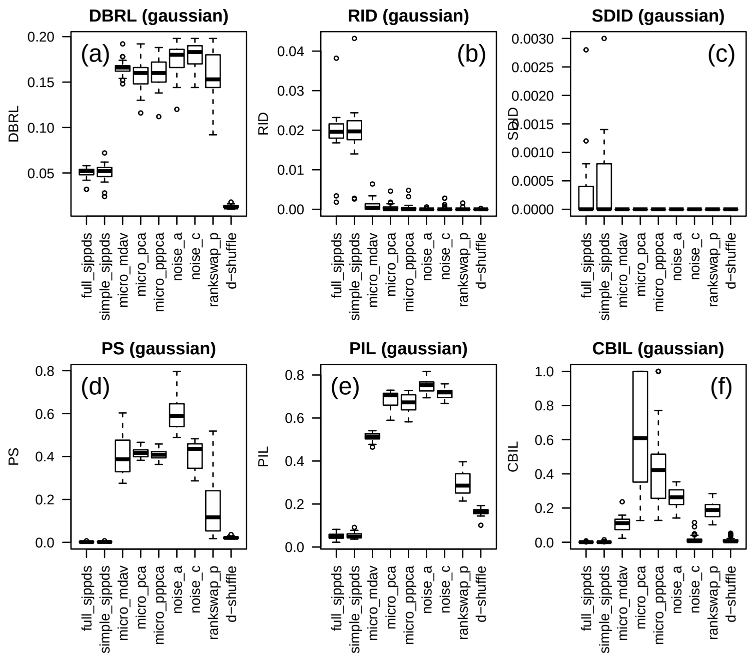

Figure 7: Metrics comparison for data perturbation methods in the simulated gaussian data experiment.

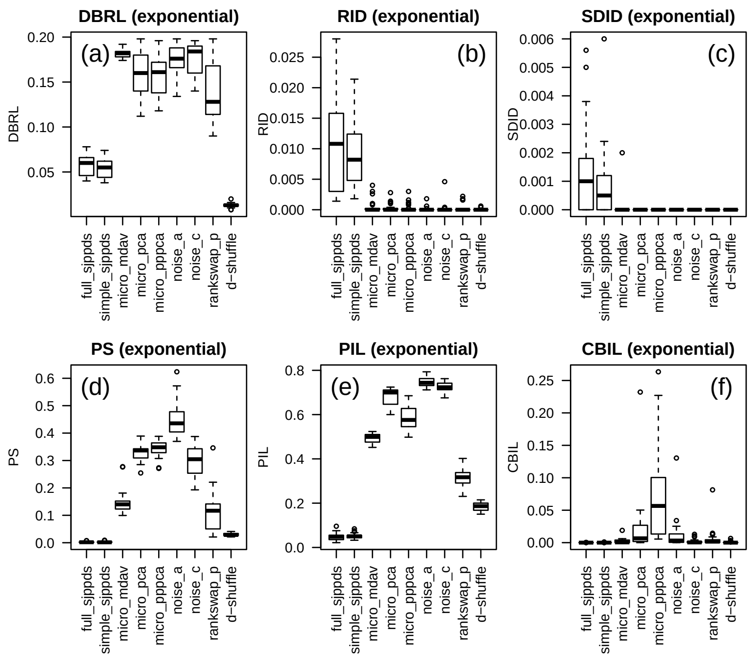

Figure 8: Metrics comparison for data perturbation methods in the simulated exponential data experiment.

plots again report the metric values based on the best selected tuning/perturbation parameter across the 30 replications of the experiments. These figures show that no single SDC method outperformed all others across all DR and IL metrics.

In terms of DBRL, Figures 5a, 6a, 7a, and 8a show that the d-shuffle and SJPPDS methods tended to outperform all other methods. In terms of RID, Figures 5b, 6b, 7b, and 8b show that the SJPPDS approaches produced higher disclosure risks than the other methods, although the disclosure risks were still fairly low (e.g., below 0.04 in all experiments). Similarly, for the SDID metric, Figures 5c, 6c, 7c, and 8c show that the SJPPDS approaches tended to produce higher disclosure risks than the other methods, although they outperformed the microaggregation methods on the Tarragona dataset (Figure 6c) and, again, tended to be fairly low (e.g., below 0.015 in all experiments).

In terms of IL metrics, panels d, e, and f of Figures 5, 6, 7, and 8 show that the SJPPDS methods tended to outperform all other methods in terms of the IL metrics, being considerably better w.r.t. the PIL metric. Note as well that for the CBIL metric (panel f), the noise addition with correlated errors method (noise-c) also tended to perform well in all datasets (although the SJPPDS still performed slightly better, as shown in Table A1 in the Appendix). The fact that the noise-c method performs well according to this metric is not surprising, given that this perturbation method actively preserves the correlation structure of the data and the CBIL metric measures how well the correlation structure is preserved.

Hence, all in all, the better overall performance of the SJPPDS approaches (Figure 4) appears to be explained by their ability to trade a small increase in attribute disclosure risk (measured by RID and SDID) by considerable lower levels of information loss (measured by the PS, PIL, and CBIL), while still maintaining a relatively low re-identification disclosure risk (measured by DBRL).

4.2 Comparisons against synthetic data approaches

We compared the performance of the proposed method against the cart approach for synthetic data generation [21], implemented with the synthpop R package [19]. We did not pursue comparisons against IPSO methods because the assumption of multivariate normality across all variables made by these approaches is rather strong, and almost never realized in real settings (it is certainly violated in the two real datasets we evaluated).

For these comparisons we adopted the same tuning parameter grid for the SJPPDS approaches as before. For the cart approach, whose performance can be influenced by the order of the variables during the data synthesis we adopted 30 distinct randomly drawn variable orderings.

For these comparisons against synthetic data methods, disclosure risk was now measured as,

$$\text{DR3} = 1 - \text{CM3} \,,$$

where CM3 corresponds to the permutation model based confidence metric for synthetic datasets proposed in [10] (and described in Appendix A1.3). As before, we evaluate the trade-off between disclosure risk and information loss using an overall score calculated as a combination of metrics,

$$\text{overall score} = \left[\frac{\text{DR3}}{2}\right] + \left[\frac{\text{PS}}{6} + \frac{\text{PIL}}{6} + \frac{\text{CBIL}}{6}\right] \,, \tag{7}$$

which gives equal weights to the DR3 metric (0.5) and combination of IL metrics (1/6 + 1/6 + 1/6 = 0.5).

Figure 9 reports the results from all experiments. The top panels show boxplots comparing the overall scores of each of the methods. As before, for the real data experiments (panels a and b), the boxplots report the results from 30 replications of the experiments based on the best selected tuning parameter, while for the simulated data experiments (panels c and d) the boxplots report the scores from the 30 distinct simulated datasets. (Again, each method is first evaluated on a grid of 30 distinct tuning parameter values, but the boxplots only report the results based on the best selected tuning/perturbation parameter.) In terms of the data confidentiality/utility tradeoff measured by the overall score, the SJPPDS approaches outperformed the cart method in all experiments.

It is interesting to notice that the SJPPDS methods tended to outperform the cart approach by a larger margin in the real datasets than in the simulated datasets (note the larger gaps between the overall scores of the SJPPDS methods relative to the cart approach in panels a and b of Figure 9 when compared to panels c and d). This could potentially be explained by the fact that the real data is considerably nosier than the simulated datasets, making it more challenging to obtain regression trees that fit well the data.

Additionally, as pointed out before, an important drawback of data synthesis methods based on fully conditional specification (such as cart) is that the quality of the synthetic data depends on the order of the variables used for the data synthesis. The SJPPDS approaches, on the other hand, do not suffer from this caveat. To illustrate this point, the bottom panels of Figure 9 compare the overall scores of the cart model computed over 30 distinct orderings of the variables, against the SJPPDS approach computed using the same distinct orderings as the cart model (and using the best selected number of classes/labels tuning parameter). The experiments clearly show that the results from the SJPPDS approaches tend to be less variable (note the narrower boxplots) than the results from the cart model.

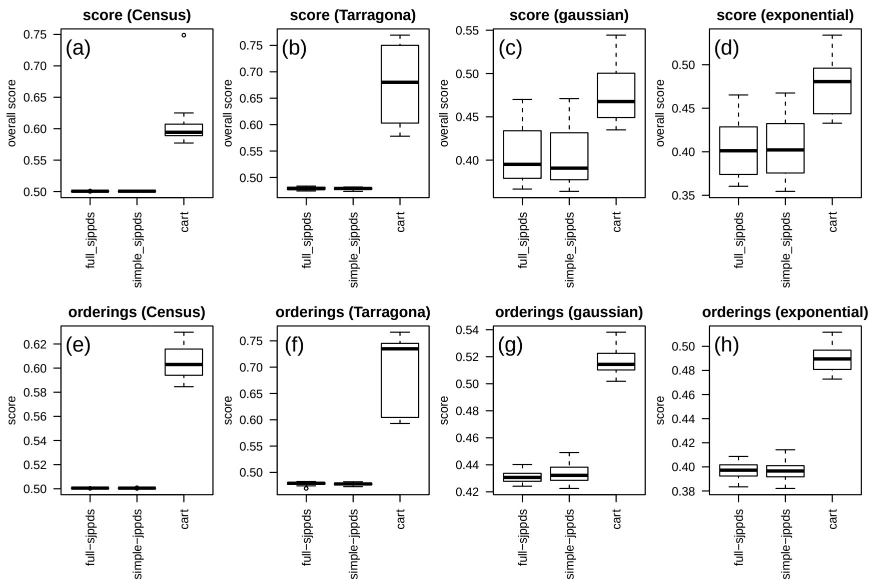

Figure 9: Synthetic data method results from the real and simulated data experiments.

Figures 10, 11, 12, and 13 present more detailed descriptions of the experimental results looking at each of the DR and IL metrics separately for each one of experiments. The boxplots again report the metric values based on the best selected tuning/perturbation parameter across the 30 replications of the experiments. These figures show that no method consistently outperformed the others for the DR3 metric (for instance, while the SJPPDS methods outperformed cart in the Tarragona data as illustrated in Figure 11a, the opposite holds true in the Census data as shown in Figure 10a). It is also important to clarify that the generally high absolute values of the DR3 metric achieved by all methods are due to the fact that the underlying CM3 metric tends to generate low values whenever the data is highly correlated and the masking methods preserve well the correlation structure observed in the original data (as pointed out in [10]). These figures also show, on the other hand, that SJPPDS methods tended to systematically outperform the cart method in terms of the three IL metrics.

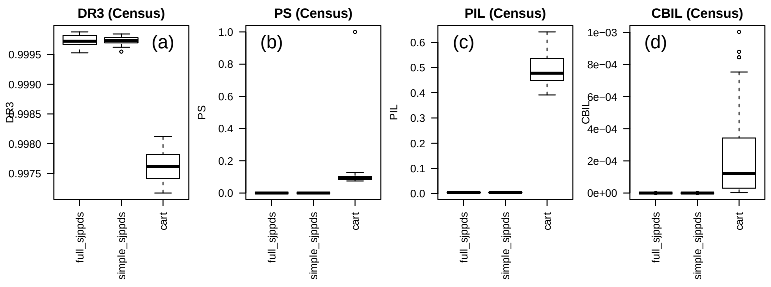

Figure 10: Metrics comparison for synthetic data methods in the Census data experiment.

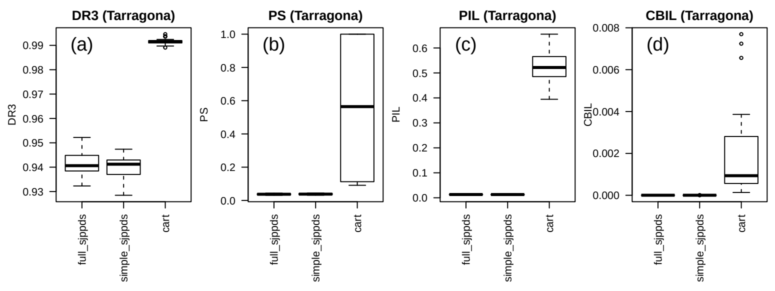

Figure 11: Metrics comparison for synthetic data methods in the Tarragona data experiment.

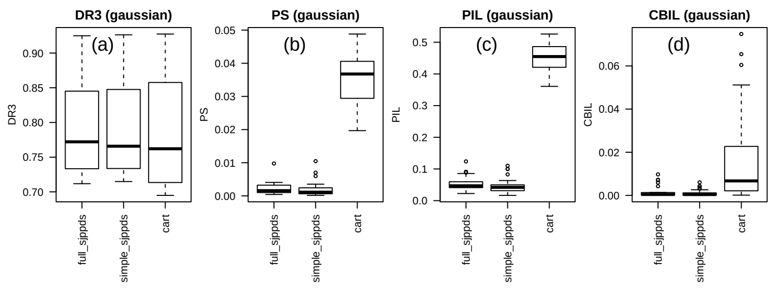

Figure 12: Metrics comparison for synthetic data methods in the simulated gaussian data experiment.

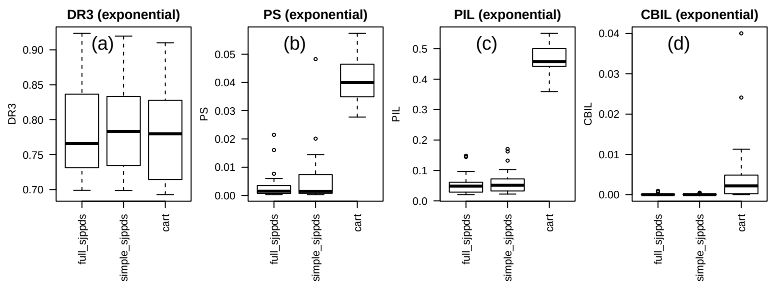

Figure 13: Metrics comparison for synthetic data methods in the simulated exponential data experiment.

5 Computation time benchmarking experiments

We evaluated the computation time of the proposed SJPPDS approaches against all other SDC methods evaluated in this paper. Each experiment was replicated 30 times, and the user times computed by the `system.time` function of R base were recorded. All experiments were performed on a Windows machine with processor Intel(R) Core(TM) i7-7820HQ CPU 2.90GHz 2.90 GHz and 64 GB of RAM. See Appendix A1.6 for further details about these experiments.

In the first experiment (Figure 14) we evaluated user time for datasets with number of records increasing from 1,000 to 10,000 and with $p = 12$ variables. Panel a shows that the full SJPPDS and microaggregation using projection pursuit principal components methods were by far the most time expensive approaches. Panel b reports the results after removing

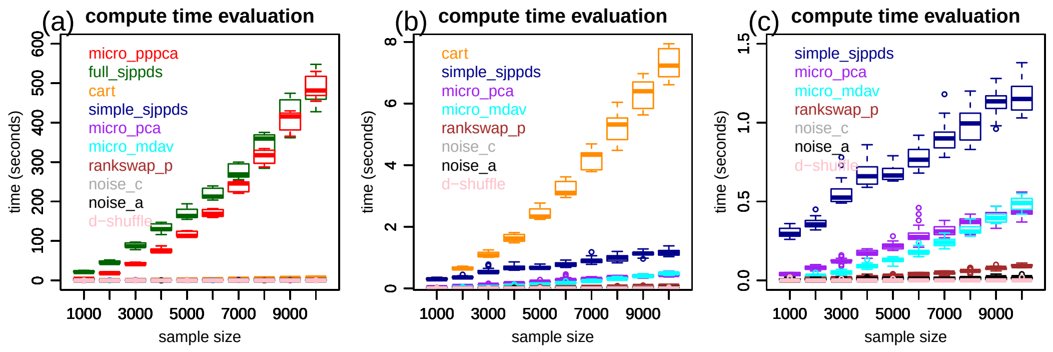

Figure 14: Computation time benchmarking of all SDC methods.

these two methods, and shows that cart is the third most expensive approach. Finally, panel c reports the results after removing the 3 most expensive approaches, and shows that the simplified SJPPDS method was the forth most expensive one. Observe, however, that the approach is still able to mask a dataset containing 10,000 records and 12 variables in under 1.5 seconds. As one would expect, the noise addition methods and d-shuffle are by far the fastest ones.

Even though we made no efforts to optimize our code for speed (it is implemented in R), overall, the simplified SJPPDS method showed competitive speed, being considerably faster than cart, but still slower than the methods implemented in the sdcMicro R package (which uses internal C++ implementations for computational efficiency). In fact, the most time consuming step in our experiments was the computation of the disclosure/information loss metrics, rather than the application of the masking methods.

These experiments also clearly illustrate that the simplified SJPPDS method should be the default choice in practice, since its computation can be orders of magnitude faster than the full version, and both approaches deliver comparable results in terms of masking performance (as illustrated by the experiment's results in the previous section).

To better understand the computational costs of the simplified SJPPDS approach we show (see Appendix A2) that the time complexity of Algorithm 3 (SJPPDS) when the joint probability preserving data shuffling is performed using the simplified version described in Algorithm 2 (JPPDS-s) is $O(n(p^2 + n_c p))$, where, as described before, n represents the sample size (number of rows), p represents the number of attributes (number of columns), and n_c represents the number of classes/labels (number of bins) parameter used in the discretization of the numeric microdata. This means that the computation time: (i) scales quadratically with the number of columns of the dataset when the number of rows and number of bins is fixed; (ii) scales linearly with the number of rows of the dataset when the number of columns and bins are fixed; and (iii) scales linearly with increases in the number of bins when the number of rows and columns are fixed.

To illustrate each of these points we performed 3 additional experiments. In the first, we set n_c to 100 and simulate datasets with $n = 10,000$ records and number of variables (p) increasing at first from 5 to 50 (Figure 15a) and then increasing from 50 to 500 (Figure 15b). These two panels clearly show a quadratic increase in computation time as p increases. In the second experiment, we set n_c to 100 and simulate datasets with $p = 10$ variables and sample sizes (n) increasing from 10,000 to 100,000 (Figure 15c), clearly illustrating the linear increase in computation time. Finally, in the third experiment we simulate datasets with

$p = 10$ variables and $n = 10,000$ records and report the computation time for number of bins (n_c) increasing from 100 to 1,000, clearly illustrating that the computation time scales linearly with n_c.

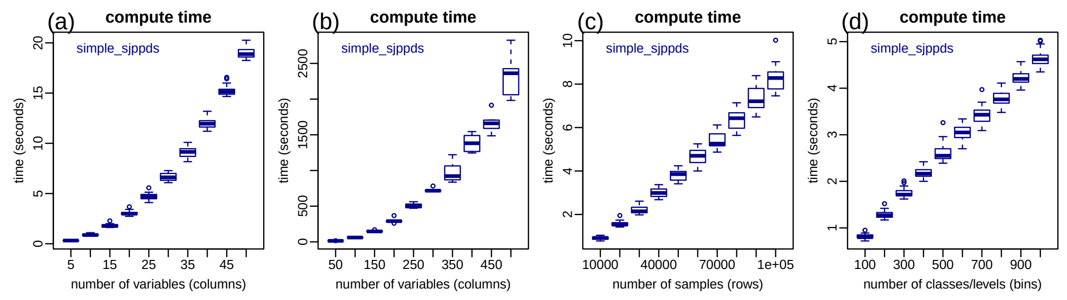

Figure 15: Compute time for the simplified JPPDS method.

6 Final remarks

We proposed a fully non-parametric and model free perturbative approach for SDC, that does not require any model specification nor is influenced by variable order. The method preserves the association structure of the data while generating marginal distributions that are identical to the original data. It is straightforward to implement and computationally light. In all experiments, it compared favorably against popular SDC approaches in terms of the confidentiality/utility tradeoff.

Due to its non-parametric and model free nature, the proposed approach might prove to be particularly effective in noisy and small datasets, which tend to be more challenging for approaches based on data modeling (which usually require good quality data in order to perform well). Quite importantly, such small/noisy datasets are likely the norm, rather than the exception, in academic research settings in health and social sciences, where preserving the confidentiality of the research subjects is key. (But, of course, the approach might also be a good option for large/high quality datasets, as well.)

While in this paper we focused on numerical variables, the approach can be directly applied to ordinal variables with high numbers of levels (which can be essentially treated as numerical variables). The approach might also be potentially extended to categorical variables with high numbers of levels, where it is possible to categorize the variables into a smaller number of levels (similarly, to the approach referred to as "generalization", "coarsening", or "global recoding" in SDC), and then perform the restricted shuffles of the original values within the levels of the coarser categorical versions of the variables.

Finally, an important limitation of the proposed approach is that it cannot be directly extended to categorical or ordinal variables with low numbers of levels, where it is not possible to further coarse the variables into fewer levels. Possible research directions to handle this situation include hybrid approaches where SJPPDS is used to mask the continuous variables and an alternative approach is used to mask the categorical ones.

R code implementing the sequential joint probability preserving data shuffling approach is available at `https://github.com/echaibub/SJPPDS`.

References

[1] Burridge, J. (2003) Information preserving statistical obfuscation. Statistics and Computing, 13: 321-327. Doi:http://dx.doi.org/10.1023/A:1025658621216

[2] Brand, R. (2004) Microdata protection through noise addition. In Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer, pages 347 - 359.

[3] Camino, R., Hammerschmidt, C., State, R. (2018) Generating multi-categorical samples with generative adversarial networks. In ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018.

[4] Cano, I., Torra, V. (2009) Generation of synthetic data by means of fuzzy c-regression. 1145-1150. 10.1109/FUZZY.2009.5277074.

[5] Domingo-Ferrer, J., Mateo-Sanz, J. M. (2002) Practical data-oriented microaggregation for statistical disclosure control. IEEE Trans. on Knowledge and Data Engineering, 14(1):189-201.

[6] Domingo-Ferrer, J., Mateo-Sanz, J.M., Torra, V. (2001) Comparing sdc methods for microdata on the basis of information loss and disclosure risk. In Pre-proceedings of ETK-NTTS 2001 vol. 2, Luxemburg: Eurostat, pp. 807 - 826

[7] Domingo-Ferrer, J., Muralidhar, K. (2016) New directions in anonymization: permutation paradigm, verifiability by subjects and intruders, transparency to users. Information Sciences, 337-338: 11-24.

[8] Domingo-Ferrer, J., Torra, V. (2001) A quantitative comparison of disclosure control methods for microdata. In Confidentiality, disclosure, and data access: theory and practical applications for statistical agencies, edited by P. Doyle, J. Lane, J. Theeuwes, and L. Zayatz. 111 - 133. Elsevier.

[9] Domingo-Ferrer, J., Torra, V. (2001) Disclosure protection methods and information loss for microdata. In Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies, P. Doyle, J.I. Lane, J.J.M. Theeuwes, and L. Zayatz (Eds.), North-Holland: Amsterdam, pp. 91 - 110, http://vneumann.etse.urv.es/publications/bcpi

[10] Domingo-Ferrer, J., Muralidhar, K., Bras-Amoros, M. (2020) General confidentiality and utility metrics for privacy-preserving data publishing based on the permutation model. IEEE Transactions on Dependable and Secure Computing, 18(5):2506-2517. DOI: 10.1109/TDSC.2020.2968027.

[11] Drechsler, J. (2011) Synthetic Data Sets for Statistical Disclosure Control. Springer, New York.

[12] Drechsler, J., Reiter, J.P. (2011) An empirical evaluation of easily implemented, nonparametric methods for generating synthetic datasets. Computational Statistics and Data Analysis, 55, 3232-3243.

[13] Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E. S., Spicer, K., de Wolf P. (2012) Statistical Disclosure Control, Wiley.

[14] Langsrud, O. (2019) Information preserving regression-based tools for statistical disclosure control. Statistics and Computing, 29, 965 - 976.

[15] Mateo-Sanz, J.M., Sebe, F., Domingo-Ferrer, J. (2004) Outlier protection in continuous microdata masking. International Workshop on Privacy in Statistical Databases. PSD 2004: Privacy in Statistical Databases pp 201-215.

[16] Mateo-Sanz, J.M., Domingo-Ferrer, J., Sebe F. (2005) Probabilistic information loss measures in confidentiality protection of continuous microdata. Data Mining and Knowledge Discovery 11(2):181-193.

[17] Moore, Jr R. (1996) Controlled data-swapping techniques for masking public use microdata, U.S. Bureau of the Census Statistical Research Division Report Series, RR 96-04.

[18] Muralidhar, K., Sarathy, R. (2006). Data shuffling - a new masking approach for numerical data. Management Science, 52, 658-670.

[19] Nowok, B., Raab, G. M., Dibben, C. (2016) synthpop: bespoke creation of synthetic data in R. Journal of Statistical Software, 74(11), 1-26. https://doi.org/10.18637/jss.v074.i11

[20] Pagliuca, D., Seri, G. (1999) Some results of individual ranking method on the system of enterprise accounts annual survey, Esprit SDC Project, Deliverable MI-3/D2.

[21] Reiter, J. P. (2005) Using CART to generate partially synthetic, public use microdata. Journal of Official Statistics, 21(3):441-462.

[22] Rosenbaum, P. R., Rubin, D. B. (1983) The Central Role of the propensity score in observational studies for causal effects. Biometrika, 70: 41-55.

[23] Templ, M. (2006) Software development for SDC in R. In: Domingo-Ferrer J, Franconi L (eds.) PSD 2006. LNCS, vol. 4302, pp. 347 - 359. Springer, Heidelberg.

[24] Templ, M., Kowarik, A., Meindl, B. (2015) Statistical disclosure control for microdata using the R package sdcMicro. Journal of Statistical Software, 67(4), 1-36. https://doi.org/10.18637/jss.v067.i04

[25] Willenborg, L., De Waal, T. (1996) Statistical Disclosure Control in Practice. Springer-Verlag.

[26] Woo, M., Reiter, J., Oganian, A., Karr, A. (2009) Global measures of data utility for microdata masked for disclosure limitation. Journal of Privacy and Confidentiality, 1: 111-124.

Appendix

A1 Experimental evaluation details

A1.1 Simulated datasets details

We performed 2 simulated data experiments. For the first, we generated data from a multivariate normal distribution, $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with distinct mean vectors and structured covariances matrices (with off-diagonal entries $\sigma_{ij} = \rho^{|i-j|}$, and diagonal entries $\sigma_{jj} = 1$). For each dataset, the mean values μ_j, $j = 1, \ldots, p$, and correlation parameter ρ were randomly sampled from $U(-3, 3)$ and $U(-0.8, 0.8)$ distributions, respectively. For the second simulated data experiment, we simulated correlated exponential random variables as follows. First, we simulate data from a multivariate normal random variable $\boldsymbol{Z} \sim N_p(\boldsymbol{0}, \boldsymbol{\Sigma})$, then, for $j = 1, \ldots, p$, we compute the correlated uniform variables $U_j = \Phi(Z_j)$, and the correlated exponential random variables $X_j = G_\lambda^{-1}(U_j)$, where Φ and G_λ represent, respectively, the cumulative distribution functions of standard normal and exponential (with rate λ) random variables. For each dataset, we randomly sample λ and ρ from $U(0.1, 10)$ and $U(-0.8, 0.8)$ distributions, respectively (and compute $\boldsymbol{\Sigma}$ as before). In each of the 2 experiments we generated 30 datasets, $\boldsymbol{X}$, of dimension $n = 500$ by $p = 10$.

A1.2 Implementation of the d-shuffle approach

As described in reference [18], in situations where all variables in the data are confidential, the data shuffling approach can still be applied as follows:

1. Estimate the Spearman correlation matrix, $\boldsymbol{R}$, from the original data $\mathbf{X}$ (of dimension $n \times p$).
2. Compute the corresponding correlation matrix, $\boldsymbol{\rho}$, from $\boldsymbol{R}$, where
$$\rho_{ij} = 2 \sin\left(\frac{\pi\, r_{ij}}{6}\right) ,$$
and r_{ij} corresponds to the ijth entry of $\boldsymbol{R}$.
3. Simulate the synthetic data set, Y^*, from a multivariate normal distribution $N_p(\boldsymbol{0}, \boldsymbol{\rho})$.
4. Obtain the reverse mapped data, Y, obtained by replacing $y^*_{(i),j}$ with $x_{(i),j}$, for $(i) = 1, \ldots, n$, $j = 1, \ldots, p$, where $x_{(i),j}$ corresponds to the ith ordered observation of the jth variable (i.e., rank$(x_{(i),j}) = i$), and $y^*_{(i),j}$ is defined in a similar way.

A1.3 Disclosure risk metrics

In our evaluations of the data perturbation methods we adopt the following DR metrics:

1. The distance based record linkage (DBRL) metric [20, 8] is one of the most widely used methods for quantifying re-identification disclosure risk in SDC. It is implemented by first standardizing the variables in the data (to avoid scaling issues) [8], and then computing the euclidean distances between each record in the masked dataset against all records in the original dataset. A masked record is then classified as "linked" when the nearest record in the original dataset turns out to be the corresponding original record. The metric is then computed as the proportion of masked records that turn out to be linked to original records.

2. The rank interval distance (RID) metric [8] is a popular metric for measuring attribute disclosure risk. It corresponds to the proportion of original records inside a rank interval whose center is the corresponding masked record. The rank interval is computed as follows. Each variable in the masked data is independently ranked and a rank interval is defined around the value that the variable takes on each record, r_{ij} (where r_{ij} represent the rank of the ith record for jth variable). The rank interval is defined as $[r_{ij} - n\,p, r_{ij} + n\,p]$, so that the ranks of values within the interval differ by less than p percent of the total number of records, n. A record in the original dataset is considered to be inside the rank interval of masked record i if, for all variables j, it is inside the respective rank interval. The interval distance is then computed as the proportion of original records inside a rank interval. Following the recommendations in [8], we report average RID values, obtained by averaging the RID for p varying from 1% to 10% (in 1% increments).

3. The standard deviation interval distance (SDID) metric [15] is another popular metric for measuring attribute disclosure risk. It is computed exactly as the RID metric, but with the exception that intervals are built around the raw values of the masked variables (rather than around their ranks), and the interval width is computed in terms of a percentage p of the standard deviation of the variable (rather than in terms of a rank percentage).

For the synthetic data methods we adopted the DR3 metric, defined as,

$$\text{DR3} = 1 - \text{CM3} \ ,$$

where the CM3 metric [10] is computed as follows:

- For $j \in \{1, 2, \ldots, p\}$:
  - Sort the original data set by its jth attribute and let $\boldsymbol{X}_s[, -j]$ be the projection of the sorted data set on all attributes except the jth one.
  - Sort the synthetic data set by its jth attribute and let $\boldsymbol{X}^\star_s[, -j]$ be the projection of the sorted data set on all attributes except the jth one.
- Compute CM3 as,

$$\text{CM3}(\boldsymbol{X}, \boldsymbol{X}^\star) = \min_{1 \leq j \leq p} \{\text{CM2}(\boldsymbol{X}_s[, -j], \boldsymbol{X}^\star_s[, -j])\} \ ,$$

  where,

$$\text{CM2}(\boldsymbol{A}, \boldsymbol{B}) = \prod_{j=1}^{p} (1 - \rho_j^2) \ ,$$

  and ρ_j, $j = 1, \ldots, p$, represents the canonical correlation between the (ranks of) $\boldsymbol{A}$ and $\boldsymbol{B}$ datasets.

Basically, the CM3 metric measures confidentiality as the minimum value of CM2 over all possible sortings of the original and synthetic datasets with respect to a single attribute. As pointed in [10] this strategy circumvents the need to know the mapping between records in the original and synthetic datasets.

A1.4 Information loss metrics

In our evaluations we adopt the following IL metrics:

1. Propensity score (PS) metric [26]. In the causal inference literature, the propensity score is defined as the probability of being assigned to treatment group, given the values of covariates [22]. When two large groups have similar distributions of propensity scores, the groups should have similar covariate distributions. In the context of SDC, one can stack the original and masked datasets, adding a variable "treatment", set to 1 for masked records and to 0 for the original records, and then compute propensity scores (i.e., the probability of being a masked record) for all records in the stacked dataset. If the distribution of the propensity scores for the original records is similar to the distribution of propensity scores of the masked records, this means that the original data is similar to the masked data and the masking method has incurred a small amount of information loss. Following [26] we estimate the propensity scores using logistic regression using a second order polynomial on all variables and including all two-by-two interactions. (The propensity scores are simply the predicted probabilities of the logistic regression model.) The similarity of the propensity score distributions is computed as,

$$\text{PS} = \frac{1}{2\,n} \sum_{i=1}^{2\,n} (p_i - 1/2)^2 \;, \tag{1}$$

where $2n$ is the total number of records in the stacked dataset and p_i is the propensity score for the ith record. Note that equation (1) will be maximal (assuming value 0.25) when p_i is either 1 or 0 for all i (in which case the original and masked datasets are completely distinguishable), and will be minimum (assuming value 0) when $p_i = 0.5$ for all i (in which case the datasets are completely undistinguishable). Because PS is bounded in the interval [0, 0.25] we multiply its value by 4 in order to obtain an information loss metric with range in the [0, 1] interval.

2. Probabilistic information loss (PIL) metric [16] quantifies information loss from a probabilistically perspective based on the differences between statistics from the original and perturbed datasets. For any population parameter θ and corresponding sample statistic $\hat{\theta}$ in the original data, $\mathbf{X}$, the PIL approach computes the corresponding sample statistic $\hat{\theta}^\star$ on the masked dataset $\mathbf{X}^\star$, and measures information loss using the standardized discrepancy statistic, $Z = (\hat{\theta}^\star - \hat{\theta})/Var[\hat{\theta}^\star]^{1/2}$. Under the assumption that Z converges to a $N(0,1)$ distribution, the PIL approach quantifies the information loss w.r.t. θ through the probability,

$$\text{PIL}(\theta) = 2\,P\left(0 \leq Z \leq |\hat{\theta}^\star - \hat{\theta}|/Var[\hat{\theta}^\star]^{\frac{1}{2}}\right) \;, \tag{2}$$

Note that $\text{PIL}(\theta)$ is 0 when there is no loss of information (i.e., $\hat{\theta}^\star = \hat{\theta}$), and increases towards 1 as the absolute discrepancy $|\hat{\theta}^\star - \hat{\theta}|$ increases. We are often interested in evaluating information loss across many distinct population parameters and following [16], we compute the average PIL across means, variances, covariances, correlations, and sample quantiles.

3. The covariance-based bounded utility metric (UM) proposed by [10], captures the similarity between the covariance matrices of the original and masked data, and is

bounded between 0 and 1. It represents an improvement over the use of eigenvalues to compare covariance matrices since it can capture both the magnitude and direction of the spread of the data over orthogonal directions. (As pointed by [10], the comparison of eigenvalues only captures the magnitude of the maximum spreads, and two datasets can have the same eigenvalue while being different. For example, if $\boldsymbol{X}^\star$ is a simple rotation of $\boldsymbol{X}$, both datasets will share the same eigenvalue.) Note that reference [10] defined this metric in terms of utility, rather than information loss, and denote it by UM. Therefore, we subtract it from 1 to convert it to a IL metric,

$$\text{CBIL} = 1 - \text{UM} \,.$$

The UM metric is computed as follows. Let $\boldsymbol{C}_{\boldsymbol{X}\boldsymbol{X}}$ and $\boldsymbol{C}_{\boldsymbol{X}^\star\boldsymbol{X}^\star}$ represent, respectively, the covariance matrices from $\boldsymbol{X}$ and $\boldsymbol{X}^\star$, and $\{\lambda_j^{\boldsymbol{X}}, \mathbf{v}_j^{\boldsymbol{X}}\}$ and $\{\lambda_j^{\boldsymbol{X}^\star}, \mathbf{v}_j^{\boldsymbol{X}^\star}\}$, $j = 1, \ldots, p$, represent the eigenvalues and eingenvectors from $\boldsymbol{C}_{\boldsymbol{X}\boldsymbol{X}}$ and $\boldsymbol{C}_{\boldsymbol{X}^\star\boldsymbol{X}^\star}$, respectively, so that,

$$\lambda_j^{\boldsymbol{X}} = (\mathbf{v}_j^{\boldsymbol{X}})^T \boldsymbol{C}_{\boldsymbol{X}\boldsymbol{X}} (\mathbf{v}_j^{\boldsymbol{X}}) \,.$$

Now consider,

$$\lambda_j^{\boldsymbol{X}^\star|\boldsymbol{X}} = (\mathbf{v}_j^{\boldsymbol{X}})^T \boldsymbol{C}_{\boldsymbol{X}^\star\boldsymbol{X}^\star} (\mathbf{v}_j^{\boldsymbol{X}}) \,.$$

The UM metric is then defined as,

$$\text{UM}(\boldsymbol{X}, \boldsymbol{X}^\star) = \begin{cases} 1, \text{ if } \lambda_j^{\boldsymbol{X}} = \lambda_j^{\boldsymbol{X}^\star|\boldsymbol{X}} = 1/p, \; j = 1, \ldots, p; \\ 1 - \min\left(1, \frac{\sum_{j=1}^{p} (\lambda_j^{\boldsymbol{X}} - \lambda_j^{\boldsymbol{X}^\star|\boldsymbol{X}})^2}{\sum_{j=1}^{p} (\lambda_j^{\boldsymbol{X}} - 1/p)^2}\right) \;, \text{ otherwise.} \end{cases}$$

A1.5 Worse case sorted disclosure risk metrics

As described in the main text, all 3 DR metrics selected for our comparisons against data perturbation methods were developed to evaluate masking methods for which there exists a mapping between the original, x_{ij}, and masked values, $x_{ij}^\star$. While this mapping is (at first sight) destroyed by the data shuffling process employed by the SJPPDS and d-shuffle approaches, in practice, it is possible to find a mapping between the original and perturbed data values using an analogous strategy as the one adopted by [10] for the computation of the CM3 metric (which was developed to measure confidentiality of synthetic data methods, where no direct mapping between original and synthetic values is available). A full description of the CM3 metric is provided in Section A1.3, but, basically, the procedure goes as follows. First, an approximate mapping between the original and synthetic datasets is obtained by sorting the rows of both the original and synthetic datasets according to the values of a given attribute (column) of the data. Second, for each data sorting, the CM2 metric (described in Section A1.3) is then computed on the sorted datasets (after the removal of the column used to sort the data). Third, the worse case (lowest) CM2 metric value across all sortings is selected as the confidentiality metric.

This procedure for the computation of the CM3 metric is justified under the permutation paradigm for data anonymization proposed by [7]. According to the permutation paradigm, the protection provided by an anonymization method results from two modifications to the original data, $\boldsymbol{X}$, namely: (i) changes in the ranks of attribute values via a permutation of the original data into its reverse mapped version (obtained by reverse mapping the original data with respect to the masked data); and (ii) potential addition of noise to the reverse-mapped data (without altering the record's ranks). Since the second step does not change ranks, the primary principle of confidentiality protection is the permutation. This means that any masked variable can always be viewed as a permutation of the variable's original values (plus perhaps a small amount of noise). Hence, as discussed in Section 3 of reference [10], even if the masked values are not directly related to the original values such as, for example, in synthetic datasets, a permutation linking the synthetic and original values always exists. Therefore, the permutation model tells us that replacing original by synthetic data can still be viewed as a permutation, and a possible approach for an intruder to find a hypothesized mapping between the original and synthetic data is to sort both the original and synthetic data by their jth attribute and, for any arbitrary jth attribute, hypothesize that the ith sorted original record corresponds to the ith sorted synthetic record.

Clearly, a similar argument can be made for the masked datasets generated by the SJPPDS and d-shuffle methods (for which the marginal distributions of perturbed datasets are identical to the marginal distributions of the original data and we can find an exact mapping between the sorted attribute j in the original data and sorted attribute j in the synthetic data for any j). Hence, a sensible way to find the mapping between the masked and original data values to compute the DBRL, RID and SDID metrics is to: (i) simply sort the original and masked datasets according to the value of one of their attributes; (ii) compute the selected metric on the sorted data; and (iii) adopt the maximum metric value obtained across all variable's sortings as the DR measure (i.e., select the worse case scenario), as described in Algorithm 4. Observe, as well, that this procedure generates more conservative DR metrics (i.e., larger disclosure risks) for the SJPPDS and d-shuffle methods than the direct calculation of the metrics in the unsorted data. Figures A2, A3, and A4 show the values of the DBRL, RID and SDID metrics computed using Algorithm 4 against the respective values obtained using the naive approach of directly computing the metrics without first sorting the data values.

In our evaluations we use the more conservative approach in Algorithm 4 for the evaluation of the SJPPDS and d-shuffle methods, but the "naive" approach for the evaluation of the microaggregation, noise addition, and rank swapping approaches.

Algorithm 4: Worse case sorted version of metric M. (M can represent any of the DBRL, RID, and SDID metrics.)

```
Data: Original microdata, $\boldsymbol{X}$; masked microdata, $\boldsymbol{X}^\star$
1 for $j$ in $1, \ldots, p$ do
2     Find the order of variable $j$ in the original dataset, i.e., $o_j \leftarrow \text{Order}(\boldsymbol{X}[, j])$, and
        sort the records of the original data (across all variables) according to $o_j$, i.e.,
        $\boldsymbol{X}_s \leftarrow \boldsymbol{X}[o_j, ]$
3     Find the order of variable $j$ in the masked dataset, i.e., $o_j^\star \leftarrow \text{Order}(\boldsymbol{X}^\star[, j])$, and
        sort the records of the masked dataset (across all variables) according to $o_j^\star$, i.e.,
        $\boldsymbol{X}_s^\star \leftarrow \boldsymbol{X}^\star[o_j^\star, ]$
4     Compute the metric using the sorted datasets $\boldsymbol{X}_s$ and $\boldsymbol{X}_s^\star$, i.e.,
        $M_j^s \leftarrow \text{Metric}(\boldsymbol{X}_s, \boldsymbol{X}_s^\star)$
5 Select the maximum of the $M_j^s$ metrics, i.e., $\tilde{M}^s \leftarrow \max_{1 \leq j \leq p}\{M_j^s\}$
Result: Return the worse case sorted metric, $\tilde{M}^s$
```

A1.6 Computation time benchmarking experiment details

In all computation time benchmarking experiments, we simulated data from a p dimensional multivariate normal distribution with mean vector $\boldsymbol{\mu} = \mathbf{0}$ and covariance matrix, $\boldsymbol{\Sigma}$, with off-diagonal entries $\sigma_{ij} = (-0.75)^{|i-j|}$ and diagonal entries $\sigma_{jj} = 1$. The tuning parameters were set to the following values: 100 for the SJPPDS methods; 7 for the microaggregation methods; 100% for the noise addition approaches; 15% for the rank-swapping method; and the original variable order for the cart approach.

A2 Time complexity of the SJPPDS algorithm

Here we describe the computation complexity of the SJPPDS algorithm (Algorithm 3) based on the simplified version of the joint probability preserving data shuffling algorithm (Algorithm 2).

Starting with Algorithm 2, note that steps 1, 2, 3, 4, 6, 10 and 11 have complexity $O(1)$. Step 5 has complexity $O(n)$ since the Unique operation has linear complexity (and is applied to a vector of length n). The number of iterations of the for-loop in Step 7 is n_c (since the number of unique values in the last column of C will be equal to the number of classes/levels used in the discretization). Hence, the complexity of all steps involved in the for-loop is $O(nn_c)$. (Since: step 8 has complexity $O(n)$; step 9 has complexity lower than $O(n)$, as idx has less than n elements and the Shuffle operation has linear complexity; and step 10 has complexity $O(1)$.) Hence, the total complexity of Algorithm 2 is $O(nn_c)$.

Now, moving to Algorithm 3, note that steps 1, 5, and 8 have complexity $O(1)$. Steps 2 and 6 have complexity $O(np)$ (since the discretization operation is linear on the number of samples n and we discretize the p columns of the data). Steps 3 and 7 correspond to applications of Algorithm 2 and have complexity $O(nn_c)$. Since the discretization and joint probability preserving operations are repeated p times, we have that the total complexity of Algorithm 3 is $O(p(np + nn_c)) = O(n(p^2 + n_c p))$.

A3 Additional tables and figures

method	Census	Tarragona	Gaussian	Exponential
full-sjppds	1.207322e-07	1.222306e-06	0.0004247213	1.911232e-06
simple-sjppds	1.140327e-07	1.880262e-06	0.0004096321	1.844499e-06
micro-mdav	1.223804e-03	5.894210e-03	0.1116734341	4.381303e-04
micro-pca	3.704366e-02	1.105832e-02	0.6088211884	6.671785e-03
micro-pppca	2.107202e-02	4.005678e-03	0.4223046141	5.660168e-02
noise-a	2.994296e-03	5.979386e-03	0.2627058322	3.222883e-03
noise-c	6.011815e-05	6.690388e-06	0.0062790947	3.595233e-04
rankswap-p	3.658983e-03	6.859038e-03	0.1882761978	1.531572e-03
d-shuffle	1.535186e-05	2.912124e-02	0.0043017796	6.068384e-05

Table A1: Median CBIL values for all methods across all datasets.

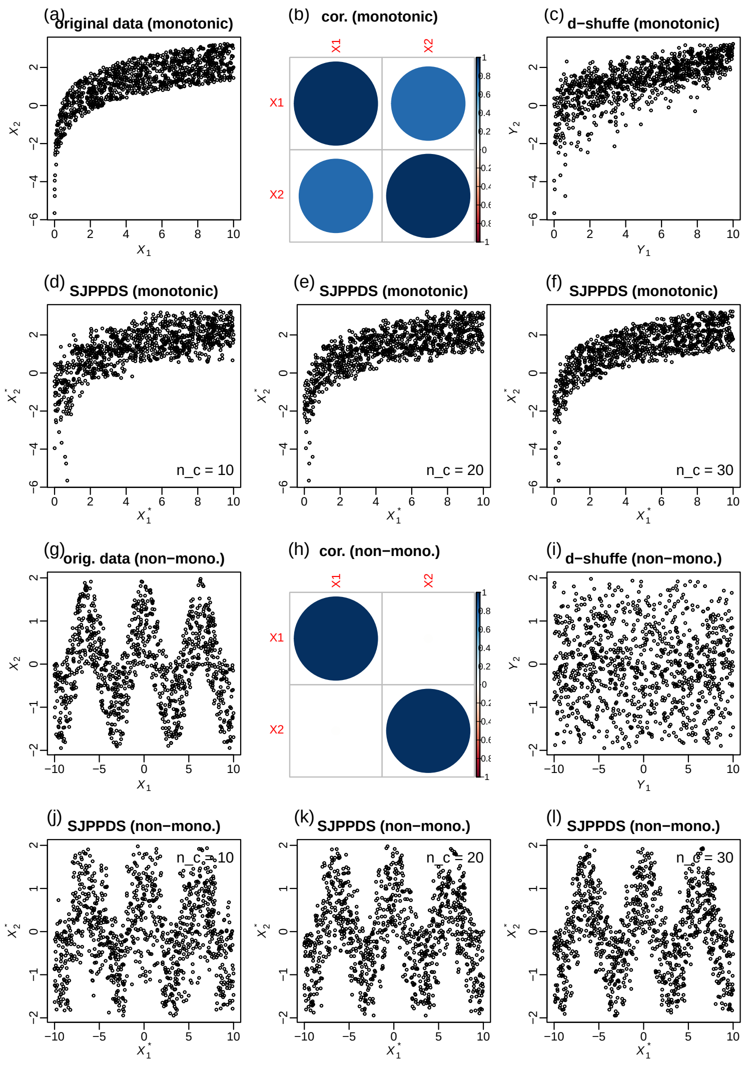

Figure A1: Comparison of the d-shuffle and SJPPDS approaches on toy artificial datasets showing strong monotonic (panels a to f) and non-monotonic (panels g to l) associations. Panel a shows a scatterplot of the original data with strong monotonic association between X_1 and X_2. Panel b presents the corresponding correlation matrix. Panel c shows the perturbed dataset generated by the d-shuffle method, while panels d, e, and f show the perturbed datasets generated by the SJPPDS method based on number of categories/levels (n_c) set to 10, 20, and 30, respectively. The d-shuffle method worked reasonably well in this example. Panel g presents a second example where the original data shows a strong non-monotonic association between X_1 and X_2, but where the correlation between these variables is nearly zero (panel h). In this case, the d-shuffle approach (panel i) is unable to preserve the non-monotonic association observed in the original data, since it relies on the estimated correlation matrix. The SJPPDS method, on the other hand, is still able to preserve this non-monotonic association, as it approximates the joint probability distribution of the original data. The data in panel a was generated by drawing 1000 i.i.d. samples according to $\epsilon_1 \sim U(0, 10)$, $\epsilon_2 \sim U(-1, 1)$, $X_1 = \epsilon_1$, and $X_2 = \log(X_1) + \epsilon_2$, while data of panel g was generated by drawing from $\epsilon_1 \sim U(-10, 10)$, $\epsilon_2 \sim U(-1, 1)$, $X_1 = \epsilon_1$, and $X_2 = \cos(X_1) + \epsilon_2$.

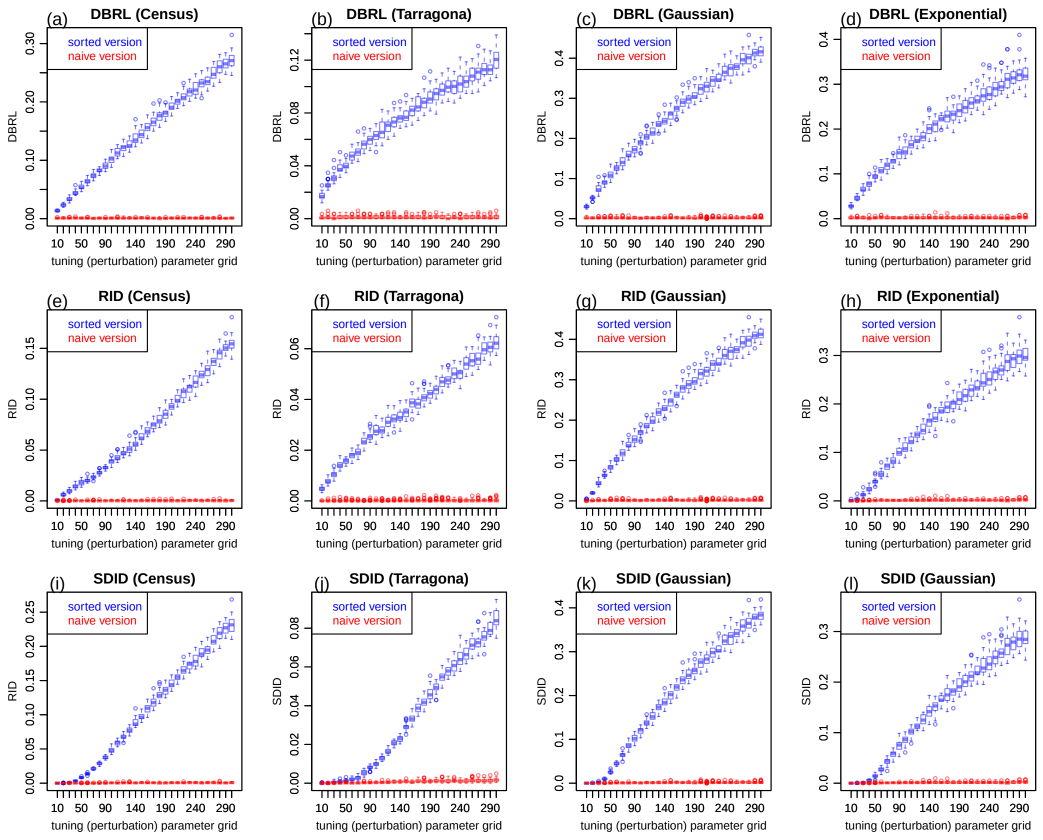

Figure A2: Comparison of disclosure metrics computed using the worse case sorted approach described in Algorithm 4 versus the naive approach of directly computing the DR metrics on the unsorted $\boldsymbol{X}$ and $\boldsymbol{X}^{\star}$ datasets. Results from the full version of the SJPPDS approach.

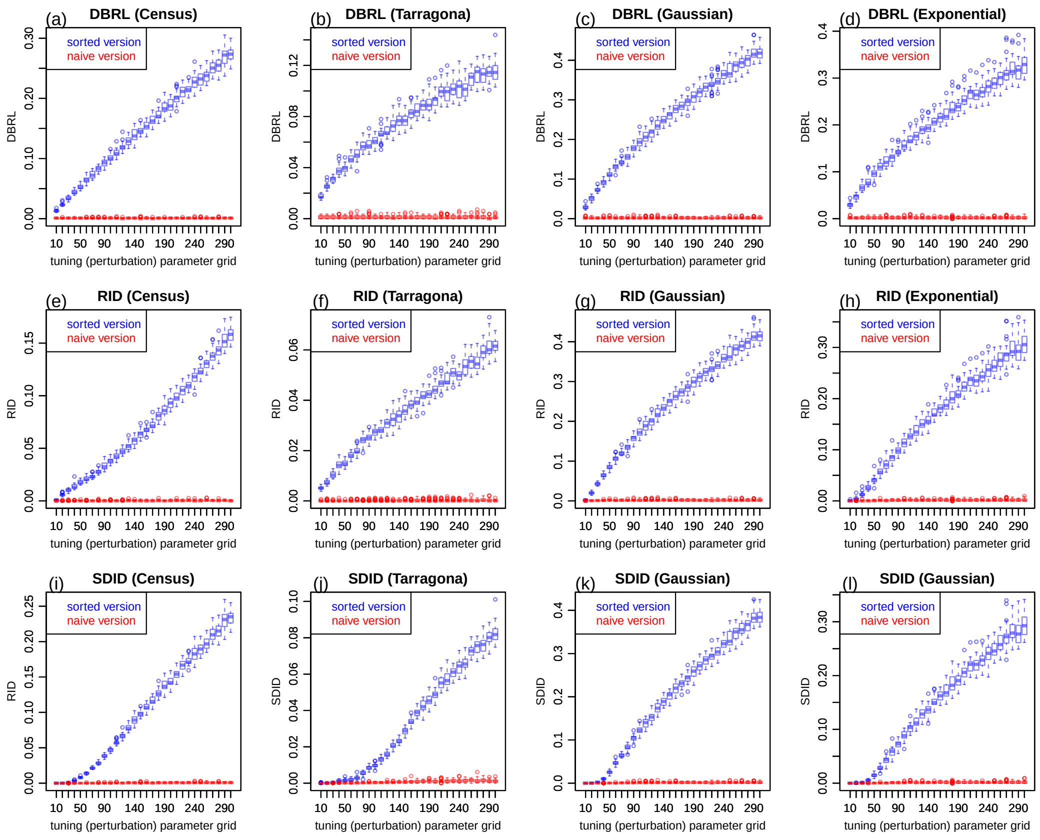

Figure A3: Comparison of disclosure metrics computed using the worse case sorted approach described in Algorithm 4 versus the naive approach of directly computing the DR metrics on the unsorted $\boldsymbol{X}$ and $\boldsymbol{X}^{\star}$ datasets. Results from the simplified version of the SJPPDS approach.

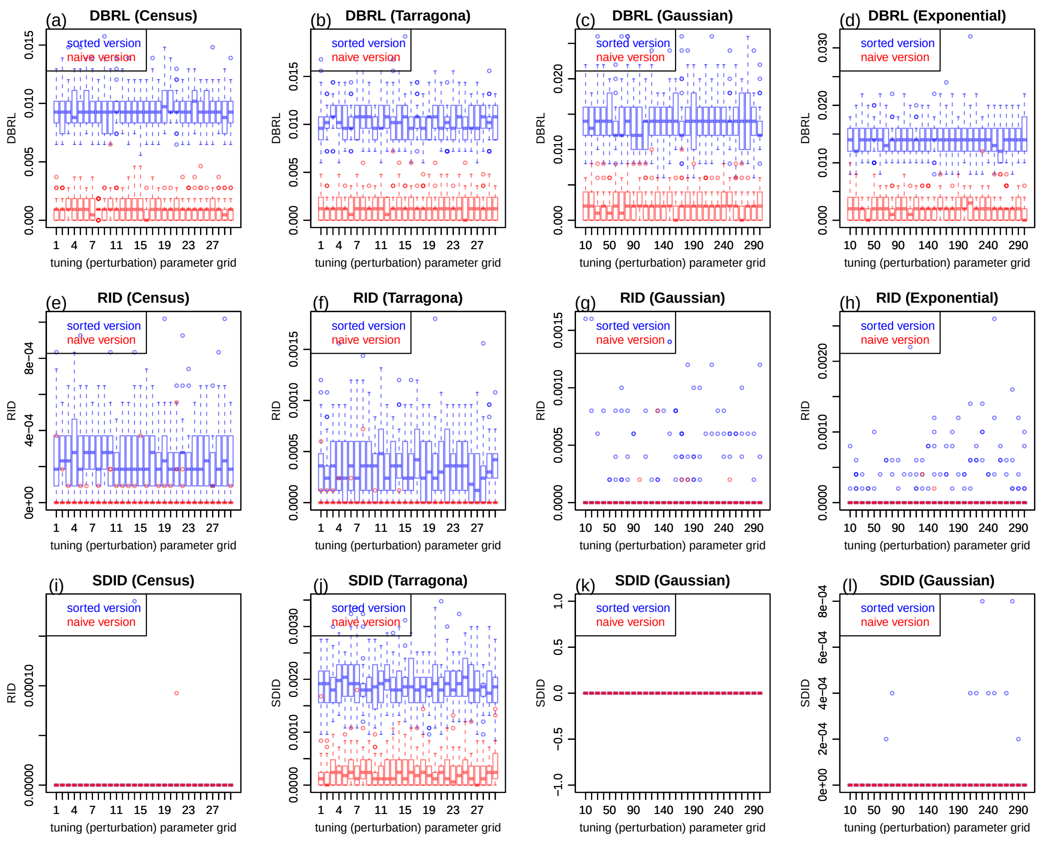

Figure A4: Comparison of disclosure metrics computed using the worse case sorted approach described in Algorithm 4 versus the naive approach of directly computing the DR metrics on the unsorted $\boldsymbol{X}$ and $\boldsymbol{X}^{\star}$ datasets. Results from the d-shuffle approach.

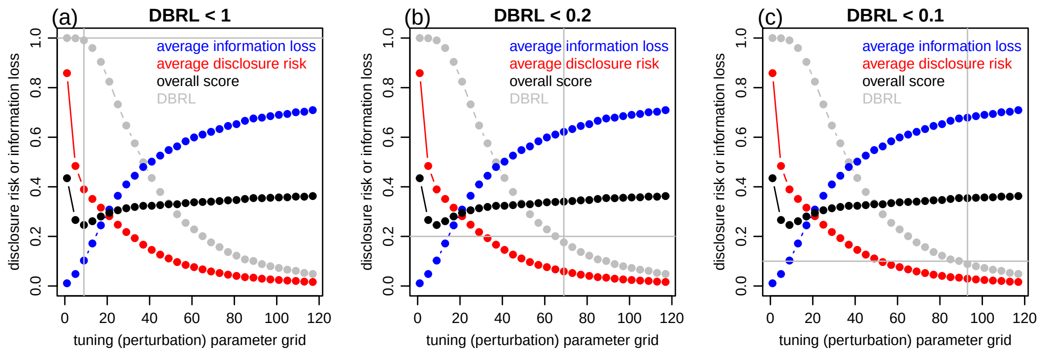

Figure A5: Selection of best tuning/perturbation parameter values conditional on different DBRL thresholds. Panels a, b, and c report the tradeoff between information loss and disclosure risk across a grid of tuning/perturbation parameter values (which in this example corresponds to the percentage of additive noise). In all panels, the blue curve represents the average information loss score (equation 5), the red curve represents the average disclosure risk score (equation 6), the black curve represents the overall score (equation 4), and the grey curve represents the DBRL metric. The grey horizontal line shows the selected DBRL threshold, while the grey vertical line shows the selected best parameter value. Panel a shows the results for DBRL threshold of 1. In this case the minimal overall score (intersection of the black curve and vertical grey line) is obtained at a perturbation parameter equal to 9% of noise. Note, however, that the corresponding DBRL value (intersection of the grey curve and grey vertical line) is close to 1 (what is unacceptably high in practice). Panel b shows the results for DBRL threshold of 0.2. In this case, we only allow tuning/perturbation parameter values for which the DBRL metric is lower than 0.2 (which is achieved at a tuning parameter value of 69% of noise). Under this restriction we have that the lowest overall score (black curve) is also achieved at 69% of noise. Panel c shows analogous results for DBRL threshold of 0.1, in which case the best tradeoff is achieved at 93% of noise.

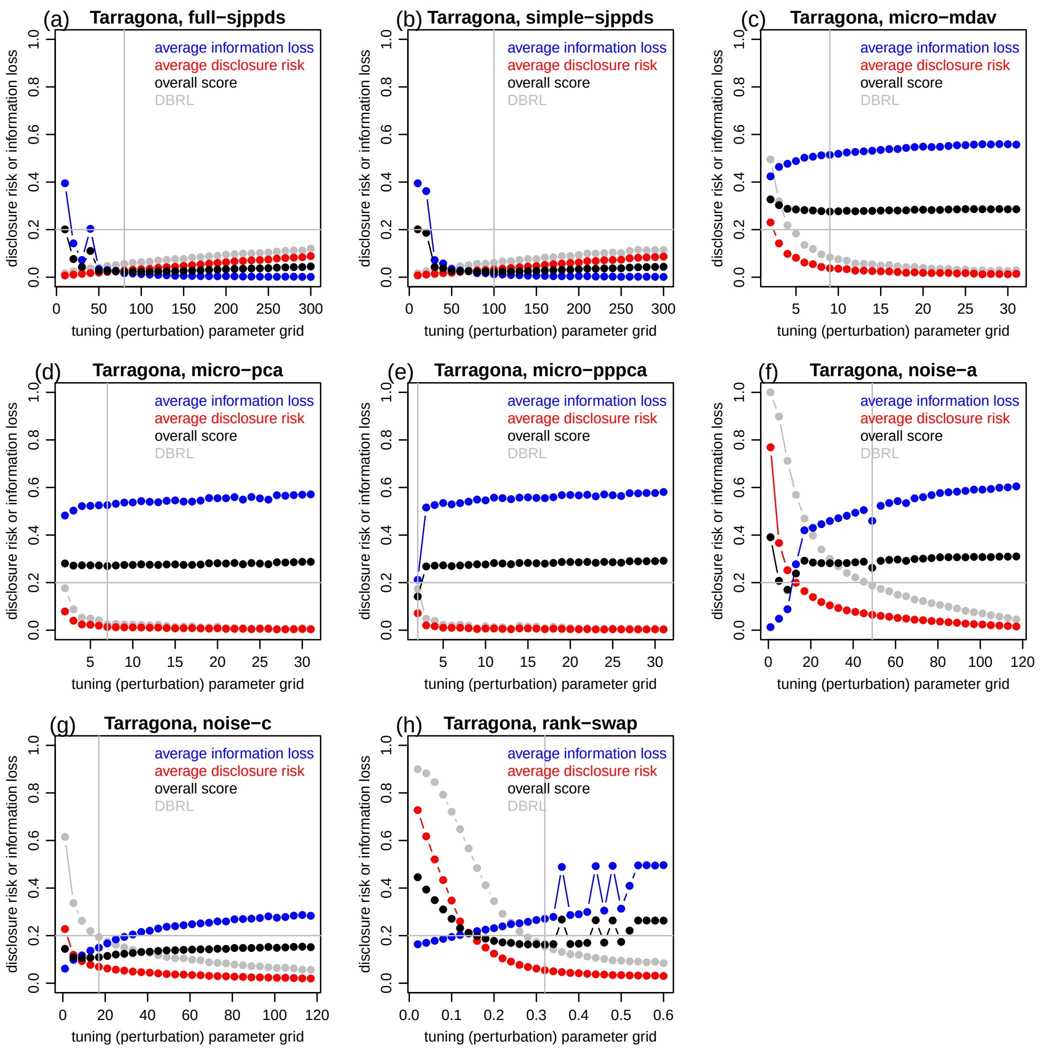

Figure A6: Tradeoff between information loss and disclosure risk in the Tarragona dataset.