UMicS: From Anonymized Data to Usable MicroData

Graham Cormode
University of Warwick
g.cormode@warwick.ac.uk

Entong Shen, Xi Gong, Ting Yu
North Carolina State University
{eshen,xgong2,tyu}@ncsu.edu

Cecilia M. Procopiuc, Divesh Srivastava
AT&T Labs–Research
{magda,divesh}@research.att.com

ABSTRACT

There is currently a tug-of-war going on surrounding data releases. On one side, there are many strong reasons pulling to release data to other parties: business factors, freedom of information rules, and scientific sharing agreements. On the other side, concerns about individual privacy pull back, and seek to limit releases. Privacy technologies such as differential privacy have been proposed to resolve this deadlock, and there has been much study of how to perform private data release of data in various forms. The focus of such works has been largely on the *data owner*: what process should they apply to ensure that the released data preserves privacy whilst still capturing the input data distribution accurately. Almost no attention has been paid to the needs of the *data user*, who wants to make use of the released data within their existing suite of tools and data. The difficulty of making use of data releases is a major stumbling block for the widespread adoption of data privacy technologies.

In this paper, instead of proposing new privacy mechanisms for data publishing, we consider the whole data release process, from the data owner to the data user. We lay out a set of principles for privacy tool design that highlights the requirements for *interoperability*, *extensibility* and *scalability*. We put these into practice with UMicS, an end-to-end prototype system to control the release and use of private data. An overarching tenet is that it should be possible to integrate the released data into the data user's systems with the minimum of change and cost. We describe how to instantiate UMicS in a variety of usage scenarios. We show how using data modeling techniques from machine learning can improve the utility, in particular when combined with background knowledge that the data user may possess. We implement UMicS, and evaluate it over a selection of data sets and release cases. We see that UMicS allows for very effective use of released data, while upholding our privacy principles.

Categories and Subject Descriptors

H.1 [**Models and Principles**]: Miscellaneous—Privacy

Keywords

Differential privacy; data release

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

1. INTRODUCTION

In the current technological environment, data is an increasingly valuable, as well as sensitive, resource. There is great sensitivity surrounding health information, location data, private communications etc., and corresponding legal and regulatory requirements to protect such data. As a result, there is a growing need to provide "anonymized" versions of data which simultaneously reveal useful information while respecting the privacy of the data subjects.

Initial efforts for developing privacy models focused on weakening (or breaking) the connection between "quasi-identifiers" and "sensitive values". However, subsequent studies showed that some of the connections can be reconstructed from the published data, using statistical inference and/or knowledge of the anonymization procedure [11, 20]. The more recent differential privacy model [6], which has gained considerable support in the research community, imposes a conceptually different condition: its output is nearly identical (in a probabilistic sense), whether or not an individual contributes his data to the set. In addition to a rigorous definition of privacy, this model enjoys several mechanisms to achieve it. The mechanisms, developed in a series of papers [7–9, 14], are often simple to implement and have great practical appeal. Thus, much of the recent work in differential privacy has focused on designing algorithms that apply these privacy mechanisms in increasingly sophisticated ways. Their goal is to preserve higher utility in the anonymized data, assuming relatively simple query workloads (e.g., range queries or the more general linear queries) [2, 12].

In this work, we offer a general solution to the effective use of privately released data, so that different models published under differential privacy can be easily incorporated and quickly benefit various data analysis tasks. Thus, our work has to carefully balance utility improvement and its applicability in practice. For example, advanced techniques have been proposed to generate synthetic data targeting at specific types of query workloads [10, 18]. Adopting such techniques may offer better utility for a certain narrow range of applications, but with poor applicability in practice due to the lack of extensibility. In the following discussion, we distinguish between the two parties involved in an exchange of anonymized data: *data owners* and *data users*. Unlike prior work, which focused mostly on designing algorithms for the data owners (while making simplistic assumptions on the data users), we focus mostly on the challenges facing data users in real-life scenarios.

Principles for privacy tool design. Most prior work [2, 12, 21] assumes a data user who only issues simple queries (e.g., range counting or linear queries) and can devote considerable resources to optimizing their answer. Recent work [19,24] studies more complex workloads, such as linear and logistic regression. In practice, workloads mix many types of queries, and data users are not able to perform query-specific result optimization. We therefore propose three principles for the design of a privacy tool.

Interoperability with existing software. Enterprise databases support many applications running various types of SQL queries, ranging from simple selections to complex joins. In this environment, a privacy tool provides a filtered view of the data to analysts who are not allowed access to the raw data. Often, these analysts already have data analysis programs running on, e.g., a prior sample of deprecated data. Thus, the data user should be able to run their queries over anonymized data that has the same (or very similar) format as the original data.

Extensibility of existing software. Most differential privacy research has focused on releasing aggregate statistics of the data, such as wavelet and Fourier coefficients [1,21]. Thus every type of query supported by the private data summary requires specialized software to map the query onto the specific model. Note that there is a high likelihood that new query types will need to be supported in the future. This will either require adapting the query-mapping tool, or violating interoperability. We adopt the guideline that the differentially private data must be able to support the same kind of applications as the original data (with some error in the results).

Scalability. Differentially private "summaries" can grow to orders of magnitude larger than the original data. Hence, the data user can incur prohibitive storage and processing costs when working with such models. We enforce scalability by requiring a new dataset having size that is similar to the original one.

These three principles support the following approach: The data user receives from the data owner some differentially private model of the original data. She then generates a synthetic dataset from this model, such that the size and format of the synthetic data are consistent with the original data. This dataset can then be used freely with existing tools and data sets. Similar approaches have been advocated in early differential privacy work [13] for masking commuting patterns in modeling geographic data. However, the approach was tailored to that specific application, and was not studied in a more general context.

Enhancing data utility. Most prior work assumes that anonymized data will be used in isolation, and so studies its utility as such. In reality, it is common that different organizations possess different pieces of sensitive information about the same individuals. For example, the employer knows one's salary, the grocery store knows one's shopping list, the clinic knows one's health condition. It is important to study how an organization can use data released under differential privacy by other organizations, given that different organizations may have overlapping information about their population. Consider the following scenarios:

Example. A hospital computes a differentially private version of the health records of its patients, including some demographic data. A researcher conducts a study over this data to find potential correlations between some demographic attributes and the sensitive diagnostic values. The researcher does not use any other dataset for this study, either as a requirement of the study or for lack of access to relevant data.

The hospital also releases the anonymized data to an insurance company. The company does not have the exact sensitive diagnosis but has access to the demographic information about its own customers (who were treated in that hospital), and would also like to do data analysis involving the sensitive diagnosis to better understand and analyze its customers. How much additional utility can the insurance company derive by combining its own database with the differentially private data released by the hospital?

In both scenarios, data users receive the same differentially private data from the data owner. However, the insurance company has access to an auxiliary dataset (its own customers) which is a subset of the data owner's population. Some attributes in the anonymized data are also present in the auxiliary table, and some are not. The data user wants to learn as much as possible about all attributes of *his own population*, without violating anyone's privacy.

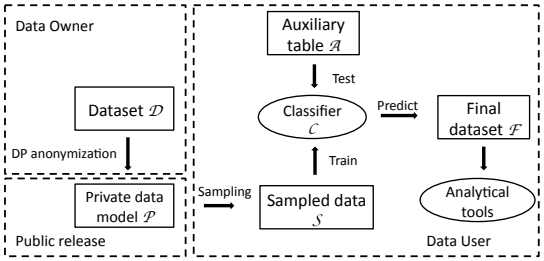


Figure 1: The UMicS workflow

We propose a framework which addresses the above scenario by combining the data owner's private release with the data user's auxiliary information. We discuss several machine learning algorithms that the data user could employ, as well as the effect of the number of common attributes the two datasets share. At one extreme, the data user has no information about any of the attributes present in the privately released data; this is consistent with the assumptions of most prior work. At the other extreme, the data user knows all but one of the attributes of the differentially private data. As we show in our experiments, the utility that the data user can derive varies significantly under these different assumptions.

Contributions. We design and analyze a privacy tool to support private data release, called UMicS, short for "Usable Micro-data Sampling". The UMicS system takes a holistic view of the practical requirements of data users, and offers an end-to-end solution to control the release and use of private data. Specifically,

- We propose a framework for synthetic data generation from differentially private models derived from real data as part of UMicS.
- We study a variety of configuration choices within UMicS that use machine learning techniques to combine the differentially private data with various types of auxiliary data, in order to improve utility;
- We evaluate possible combinations of techniques that the data owners and data users can employ within UMicS, and conclude that the data user can obtain significant utility from the released data, while making no changes to existing tools and systems.

2. OVERVIEW OF UMicS

Our UMicS system operates in a "publishing" mode: the data owner builds and releases a *private data model* that describes the distribution of data in the original dataset. We adopt the model of differential privacy—for more details, see the original papers or surveys [6]. UMicS incorporates a synthetic data generator that ensures that the resulting data is compatible with existing tools and programs. One can sample multiple synthetic datasets from the released model without decreasing the privacy level, since this is just post-processing computation over a differentially private model.

The UMicS system applies to a common scenario in data release through the following sequence of steps, illustrated in Figure 1.

1. Dataset $\mathcal{D}$. A *data owner* (DO) has an original dataset $\mathcal{D}$ which contains microdata (records on individual level) regarding each individual's demographic information. For example, these demographics can include data on the individual's age, education level, home zip code, and so on. The information about an individual can also include information which may be less widely known, to be

considered particularly private e.g., salary and disease status. We denote by 'target attribute' any attribute of $\mathcal{D}$ that is not widely known, over which a data user may wish to run some data analysis software, and for which there are significant privacy concerns. However, we emphasize that in our privacy model all attributes are potentially private, and so all are protected by the UMicS system.

2. Private data model $\mathcal{P}$. In order to share data in a privacy-preserving manner, the data owner creates a *private data model* $\mathcal{P}$ under differential privacy. In general, $\mathcal{P}$ will be a noisy description of the data distribution in some fashion. Depending on the anonymization approach chosen by DO, $\mathcal{P}$ may have different formats, e.g. a set of contingency tables or a spatial decomposition tree.

3. Private data sample $\mathcal{S}$. The *data user* (DU) is the entity who wants to do data analysis based on the private data summary $\mathcal{P}$. The data user prefers data in its original format (microdata) since most of his data analytic tools are off-the-shelf, i.e., they may not be able to take $\mathcal{P}$ directly as input. Consequently, the data user will not run queries directly over $\mathcal{P}$, but instead will generate a synthetic dataset $\mathcal{S}$, based on $\mathcal{P}$. Queries can then be directly answered over $\mathcal{S}$, satisfying the interoperability and extensibility.

4. Private classifier $\mathcal{C}$. To improve accuracy and make full use of the synthetic data $\mathcal{S}$, the data user can perform additional post-processing of $\mathcal{S}$ to obtain a richer model. Specifically, we advocate using $\mathcal{S}$ to train a classifier $\mathcal{C}$, which can learn the correlation between the target attribute and the demographic attributes.

5. Auxiliary table $\mathcal{A}$. As described in Section 1, the data user may sometimes have access to an *auxiliary table* which contains a subset of the attributes from $\mathcal{S}$. Moreover, the auxiliary table may contain a proper or partly overlapping subset of the tuples present in the data owner's original set $\mathcal{D}$. Note that while $\mathcal{P}$ is released to the public, $\mathcal{A}$ is only available to DU, i.e., $\mathcal{A}$ does not compromise the privacy of $\mathcal{P}$.

6. Final data $\mathcal{F}$. If DU has an auxiliary table $\mathcal{A}$, he can use the classifier $\mathcal{C}$ in combination with $\mathcal{A}$: for each tuple $\tau \in \mathcal{A}$, DU applies $\mathcal{C}$ to obtain predictions for the target attribute value of τ. Drawing from the distribution of these predicted values for each tuple $\tau \in \mathcal{A}$ yields a final data set $\mathcal{F}$, over which DU applies his queries of interest. If DU does not have an auxiliary table, then $\mathcal{F}$ is the synthetic dataset $\mathcal{S}$.

3. DATA OWNER

3.1 Private Data Models (Step 2 in Section 2)

We make the observation that much of the recent work in private data release can be thought of as using the input data set to obtain the parameters of a data model, and then adding suitable "noise" to these parameters. This is done so that the noisy model description meets the differential privacy definition and thus can be released. The idea of working with models is that the parameters are less sensitive to any one individual, and so noisy parameters can be quite faithful to their true values.

Below, we describe some models that are well-suited for UMicS. Because of the scalability principle, we are interested in models that are sufficiently compact, have small parameter sensitivity, and allow fast synthetic data generation.

One-way Marginals. A first model of a dataset $\mathcal{D}$ is to describe the distribution of each attribute of $\mathcal{D}$ in isolation. That is, for each attribute $\mathcal{D}.A$ (e.g., gender), we compute its marginal distribution of values (e.g., the number of males, resp. females, in the data). For simplicity, we assume that this distribution is discrete. This follows immediately when $\mathcal{D}.A$ is categorical. For continuous attributes, the distribution can be given as a histogram over $\mathcal{D}.A$. For example, we might break ages into ranges of ten years (0-9, 10-19, 20-29 etc.). There has been substantial work around how to choose the bucket boundaries for such histograms in a privacy-preserving fashion: this can be done via private quantiles for equi-depth histograms [2]; by private dynamic programming [23]; or by exact computation on similar data [16].

To release one-way marginals privately, noise is added to each entry, which is scaled by $\frac{d}{n}$, where d is the number of attributes in the data, and n is the number of individuals. Often, $d \ll n$, so the amount of noise is small. However, the model is limited: it essentially treats all attributes as if they are independent, since it does not describe any correlations between attribute values.

Contingency Tables (Multi-way Marginals). A contingency table gives the (joint) distribution of a subset of attributes, thus encoding the correlations between these attributes. E.g., a contingency table could record the joint distribution between age and gender. As in the case of one-way marginals, when an attribute is continuous, its domain is usually coarsened by imposing a grid on it. To comply with differential privacy, noise is added to the count in each cell.

Contingency tables have been advocated as a data model for many years [5]. The most natural way to release a contingency table is to directly compute and add noise to each cell independently. However, one can also compute a transformation of the table (such as a Fourier or Wavelet transform), and add noise to the coefficients of the transform. The released table is then found from the noisy coefficients [1, 21]. Separately, there is the question of which subsets of attributes to release in the form of contingency tables [4, 22]. In particular, [22] releases a d-dimensional frequency matrix, which is a full contingency table over all attributes, while [4] releases a set of contingency tables (also called *cuboids*) whose subsets of attributes may overlap. Both methods aim at minimizing the maximal noise when a range query touches a large number of cells.

Private Spatial Decompositions. Contingency tables can be seen as providing a description of the density of the data across a set of attributes. However, the density can vary within the dataset, meaning that the description of the data is too coarse in some places, and too fine in others. To remedy this, spatial decompositions have been proposed which adaptively form a description of the data based on the observed density. These begin with the full data space, and progressively partition the space into smaller regions. Finally, the density of points within each 'leaf' region is reported (with noise).

The main variation across different decompositions is in the partitioning step: how to choose which region to partition next, and how to do the partitioning. The splitting can be data-independent (splitting mid-way along each dimension, similar to quad-trees, oct-trees and binary spatial partitionings), or data-dependent (splitting based on medians, similar to kd-trees) [2]. The choice of which region to split can also be based on which region will benefit most from this refinement [15]. Other design decisions to fix include when to stop splitting, and what post-processing to do on the structure. The resulting private data summary $\mathcal{P}$ is the leaf regions of the spatial decomposition: For each leaf, we publish its boundaries (its range along each attribute), and information about the distribution within the cell, such as the count or density of points.

4. DATA USER

Once the private data model has been computed, the data owner is free to release it, knowing that the desired privacy definition has been met. Throughout this section, we consider a variety of approaches that DU can follow to answer queries: the *local*, *sample*, *hist* and *predict* methods. We explain each in turn, as they arise.

4.1 Sampling from model (Step 3 in Section 2)

In principle, the data user can adapt their queries to operate on the model in terms of its parameters, and derive query answers from them. For example, a range query can be applied to a private spatial decomposition, by finding the regions of the decomposition that it intersects, and estimating the intersection size of the query with each of these. This is the approach adopted in [2]. We refer to this as the *local* approach to query answering.

However, the data user may have difficulty making use of this model per se: it describes the data, but it is not in the form of the original microdata. It is unclear, for example, how to perform linear regression on data described in the form of a private spatial decomposition. In general, most data users prefer processing their queries directly over microdata, rather than having to map the queries into possibly complex functions of the model parameters. In UMicS, we advocate the idea of *sampling* from the data model in order to generate microdata. While the sampling step does require some effort on the part of the data user, it is simple to implement and provides universal results: once the sample is generated, all queries can be processed over it directly without additional code. The models described in Section 3 all provide a description of the density of the data in different regions, so we can adopt appropriate sampling procedures for each model in turn.

Marginals. To sample from the (noisy) marginal distributions, DU draws tuple values independently from their corresponding attribute distributions. That is, for each attribute i, DU samples a value from the marginal distribution of attribute A_i, then concatenates these values to obtain the output tuple.

Contingency Tables. The data user can similarly sample from noisy contingency tables. When there is a single contingency table that describes the joint distribution of all attributes, this can be sampled directly. Similarly, when there are multiple contingency tables, each of them over disjoint subsets of attributes, the data user can sample from each table independently and concatenate the results to generate a tuple consistent with the model. The challenging case is when there are contingency tables on overlapping subsets of attributes. In that case, the approach is to first combine these tables to generate a single contingency table over the union of their attributes, so that the result is most consistent with the data model. Various consistency-achieving methods have been suggested for this problem [1, 3, 4].

Private Spatial Decompositions. The output of a private spatial decomposition is a set of leaf regions and associated counts. The data user first samples a region with probability proportional to its noisy count. Then a sample tuple is drawn from the region by choosing uniformly among all tuples covered by the leaf region. The output set $\mathcal{S}$ is thus consistent with the data model.

4.2 Query answering in UMicS

Given a set of queries over the data, DU can directly evaluate them on $\mathcal{S}$ to get an answer. We call this the *sample* approach to query answering, and evaluate it in our experimental study. Below, we discuss ways to improve over this method.

Noisy histogram. In this approach (called *hist*), DU can project $\mathcal{S}$ on t, and obtain its (noisy) marginal distribution, $\mathcal{S}.t$. DU can then sample a value from $\mathcal{S}.t$ for each tuple in the table $\mathcal{A}$, and then apply its queries to this new table. This takes advantage of the fact that $\mathcal{S}$ gives a fairly accurate description of the global distribution of t, but does not exploit any of the information in $\mathcal{S}$ about the correlations between other attributes and t.

Combine with the user's model. We saw in Section 3 that it is common for the data owner to build a model of the data (possibly implicitly) prior to release. We now argue that it is beneficial for the data user to build their own data model from the released data. The intuition is that the released data contains information about the overall distribution spread throughout it. If the data user answers queries directly over $\mathcal{S}$, we miss out on exploiting this information.

We propose to build models of the data based on classifiers drawn from machine learning (Step 4 in Section 2). That is, the data user uses $\mathcal{S}$ as the training data for a classifier $\mathcal{C}$, with the goal of predicting the target attribute. Here, we can use an off-the-shelf classifier, such as SVM, Random Forest, Naive Bayes and so on. The training process does not need to take account of the data model used by the data owner, or the process used to generate it: we can treat $\mathcal{S}$ as standard training input to the classifier. The data owner can use $\mathcal{C}$ to help answer queries. Specifically, in the case where the data user has an additional table $\mathcal{A}$ describing a subset of individuals of interest, DU can take each tuple in table $\mathcal{A}$ and apply $\mathcal{C}$ to obtain a predicted value of target attribute t. DU can then answer queries over this data set, $\mathcal{F}$ (Steps 5 and 6 in Section 2).

While this works quite well in practice, there is some loss of information: most classifiers output not just a single predicted values, but rather a belief distribution B over values. We found that we can achieve improved accuracy if we sample from the distribution B, rather than just pick $b = \arg\max_b \Pr[B = b]$, the most likely value. This is equivalent in the limit to weighting all possible values of t by the corresponding probability from B. We call this approach to query answering the *predict* method.

5. EXPERIMENTAL STUDY

5.1 Experimental Environment

Experiments were performed on two real data sets containing demographic data: the *Adult* dataset from the UCI Machine Learning repository[1] and 2010 *Census* microdata extracted from IPUMS USA [17] The attributes we selected are summarized in Figure 2. The target attribute used for the experiment is marked with 'TA', while the rest are demographic values which may be present in the data user's table $\mathcal{A}$. After removing tuples with missing values, there are a total of 30,162 tuples in *Adult* dataset and 200,000 tuples from the *Census* dataset.

The data owner DO holds the dataset $\mathcal{D}$ while the data user DU has a table $\mathcal{A} \subset \mathcal{D}$ whose distribution may be different from $\mathcal{D}$. We experimented with a variety of distributions for $\mathcal{A}$. The results shown here are for a table $\mathcal{A}$ constructed so that records in $\mathcal{D}$ having higher attribute values are more likely to be sampled than those having lower values. In order to study the impact of dimensionality, we created d dimensional datasets for $d = \{2, 3, 4, 5\}$ by selecting the first $d - 1$ attributes and the target attribute from each of the two datasets described in Figure 2.

Measures. We study combinations of approaches by DO and DU in our UMicS workflow by comparing the accuracy of answering a variety of queries on the final dataset $\mathcal{F}$. The default workload consists of 1,000 range count queries of the form:

[1] http://archive.ics.uci.edu/ml/datasets/Adult

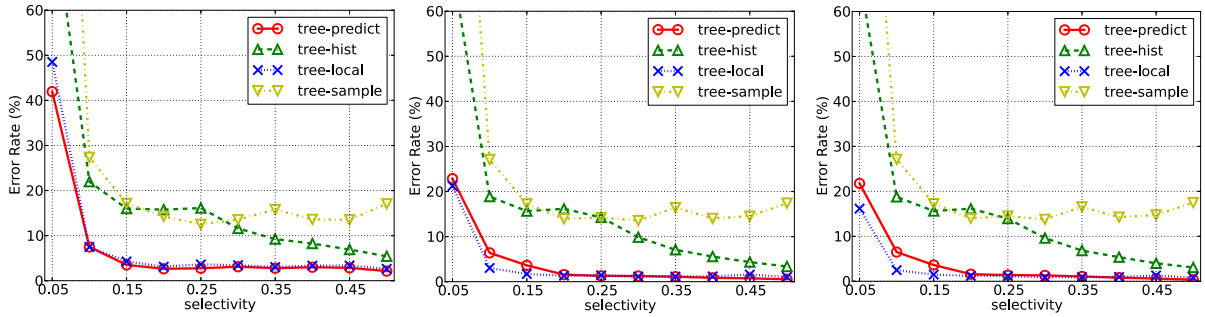
Attribute	Cardinality
education	16
age	74
marital status	7
workclass	7
occupation (TA)	14

(a) Adult dataset

Attribute	Cardinality
education	11
occupation	26
age	79
health insurance	2
salary-class (TA)	11

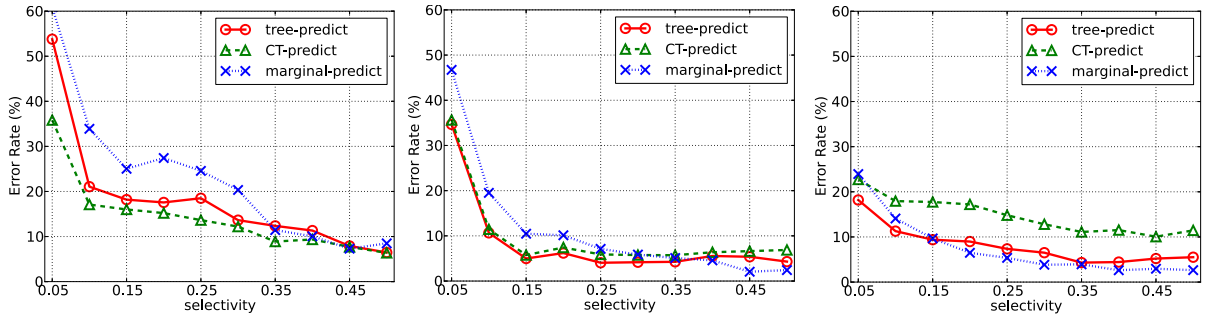
(b) Census dataset

Figure 2: Summary of dataset attributes



(a) $\varepsilon = 0.1$ (b) $\varepsilon = 0.5$ (c) $\varepsilon = 1.0$

Figure 3: DU's Choice of query strategy. *IPUMS* dataset, 3D.



(a) *Adult* 2D (b) *Adult* 3D (c) *Adult* 4D

Figure 4: DO's choice of private data model. *Adult* dataset, $\varepsilon = 0.5$.

`SELECT COUNT(*) FROM` Microdata
`WHERE` $pred(Attr_1)$ `AND ... AND` $pred(Attr_M)$ `AND` $pred(TA)$

since counting queries are the building blocks of many advanced data analysis tasks. The *selectivity* of a query q is the fraction of tuples in DU's dataset $\mathcal{A}$ that satisfy all the predicates in q (including predicates over the withheld target attribute value). Here, the generated data set (including the target attribute) is used as the ground truth for query accuracy. Since in practice DU has only the auxiliary table without the TA, the experiments measure the ability of DU to use the UMicS methodology to obtain accurate answers to queries that span the target attribute. Query accuracy is measured by the average relative error $\sum_{q\in\mathcal{Q}} \frac{|q(\mathcal{F})-q(\mathcal{A})|}{\max\{C,q(\mathcal{A})\}}/|\mathcal{Q}|$, where $q(\mathcal{F})$ and $q(\mathcal{A})$ is the query result and true result respectively for query q. We include the constant C as a sanity bound to mitigate the effects of the queries with extremely small selectivities. This is consistent with prior work that has faced similar issues in evaluating query accuracy, e.g. [21]. In the experiments we set C to be 0.1% of the total number of records. All experiments were conducted on a 3.00GHz CPU with 8GB RAM, so the data sets fit easily in memory. The reported experimental results are the average of 10 runs. We implemented our instantiations of UMicS in Python 2.6 with the scientific package Numpy to assist data handling. Three typical values of the privacy parameter ε ($\varepsilon = 0.1, 0.5, 1.0$) are used throughout, to study the impact of the privacy budget on accuracy.

5.2 DU's Choice of Query Strategy

We compare strategies DU may employ in the UMicS system. For these, we fix the choice of the private data model $\mathcal{P}$ as a private spatial decomposition tree (Section 3.1) and evaluate the four approaches of the data user introduced in Section 4: *predict, hist, local* and *sample*. Figures 3 shows the query accuracy in terms of average relative error on 3D *IPUMS* datasets for different ε values, as query selectivity is varied.

Figure 3 shows *tree-predict* and *tree-local* are the preferred query strategies for DU, providing single-digit relative error for all queries with selectivities greater than 10%. Thus, both strategies are able to capture the association between the target attribute and the other attributes. In particular, since the *local* approach requires querying directly over the private model $\mathcal{P}$ and is not always available, the *predict* method should be the top choice of DU. Moreover, in most cases a classifier can make better use of the sample data $\mathcal{S}$ than *hist* by learning and predicting instead of relying on the global distribution. We see that without the auxiliary table, using the *sample* data $\mathcal{S}$ provides limited insights in understanding DU's own population, giving much larger query errors than other approaches. This highlights the improvement possible when a data user brings some information about their own data to the analysis.

5.3 Choice of Data Model with $\mathcal{A}$

In this section we contrast three models that fit naturally within the UMicS system as discussed in Section 3.1. We fix DU's choice of query strategy as *predict*. Figure 4 shows the relative query error under different dimensionality when DO releases a spatial decomposition tree (*tree-predict*), a full contingency table (*CT-predict*) or the marginals (*marginal-predict*) on the *Adult* dataset given a certain value of $\varepsilon = 0.5$.

First, it is noticeable that no model is able to win hands down across the board. This indicates that each model has its own advantage over certain types of data (based on distribution and dimensionality). Specifically, for the contingency tables model, when

ε	original	*CT-predict*	*CT-hist*	*CT-sample*
0.1		26939	28427	27027
0.5	26784	26936	28426	27024
1.0		26933	28427	27024

Figure 5: Sum of Squared Error of Linear Regression Models

there are significant numbers of points in most cells, the independently added noise does not perturb the signal much. This explains the better query accuracy of *CT-predict* on the 2D *Adult* dataset in Figure 4(a), where there are fewer cells and so the counts in most cells are quite high. Since the *Adult* dataset has similar domain ranges for each attribute compared to *IPUMS* but far fewer records, *CT-predict* seems to lose its edge as we increase the data dimensionality, as shown in Figures 4(b) and 4(c). We explain this due to the smaller signal-to-noise ratio in each cell.

The *marginal* model drops much information about the original dataset by releasing only the noisy histogram of each attribute independently. Thus it provides less accurate query answers in most scenarios, which is in line with expectations. An exception is in Figure 4(c), where *marginal* outperforms the tree model for selectivities greater than 15%. A plausible explanation is that in *Adult* 4D data, the correlation between t and other attributes is rather weak, so the information loss in releasing the marginals is minimal, even smaller than the impact of the noise in the decomposition tree. That is, the classifier used can build a sufficiently good model of the data from just the marginal distributions for this data.

5.4 More Complex Data Analysis.

An important factor in designing UMicS is that the DU often prefers data in its original form, as it enables more advanced data analysis using off-the-shelf data mining tools, beyond simple range count queries, such as linear regression. We used standard tools to build ordinary least squares estimators on various datasets obtained by different strategies, and measure the sum of squared residuals (SSR) in DU's data. Figure 5 shows the SSRs of linear models obtained by *CT-predict*, *CT-hist* and *CT-sample* with different privacy budgets on 3D IPUMS data. Here, *salary class* is used as the dependent variable and the explanatory variables are *age* and *education*. It can be seen that *CT-predict* has the least SSR among the three approaches, which means it is able to provide a more accurate linear model, and only 0.5% greater than the SSR of the regression on the original data, indicating that data utility is well preserved in our UMicS framework for this data analysis task. This computation is also quite robust: the results change little as ε varies. Other experiments, such as more complex SQL queries, showed similar results, and are omitted for brevity.

6. CONCLUDING REMARKS

The vast interest in private data release, along with the many strong motivations for releasing data, mean that there is great pressure to enable releases to take place smoothly. We have shown the need for additional tools and systems to support the *usage* of such outputs. Since data users have existing tools and data, it is vital to enable the smooth integration of private data with these. We have proposed UMicS as a model system for permitting this data usage. We have shown how it upholds the principles of interoperability, extensibility and scalability. Our experimental results show that queries over private data can be answered effectively, with low error. Existing background knowledge of the data user can be combined with private data to enhance utility while preserving privacy.

The UMicS workflow is flexible and general. Different choices of private data model for DO, and of the data model used by DU, can be easily incorporated. The next steps are to extend the workflow to additional data release settings beyond the central case in data release with a single private table of interest. Of particular interest is the case where there is a database of private data, with multiple tables, and join relationships among them.

7. REFERENCES

[1] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, 2007.

[2] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially Private Spatial Decompositions. In *ICDE*, 2012.

[3] G. Cormode, C. M. Procopiuc, D. Srivastava, and G. Yaroslavtsev. Accurate and efficient private release of datacubes and contingency tables. In *ICDE*, 2013.

[4] B. Ding, M. Winslett, and J. Han. Differentially private data cubes: optimizing noise sources and consistency. *SIGMOD*, 2011.

[5] G. Duncan, S. Fienberg, R. Krishnan, R. Padman, and S. Roehrig. Disclosure limitation methods and information loss for tabular data. In *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, 2001.

[6] C. Dwork. Differential privacy. In *ICALP*, 2006.

[7] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006.

[8] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography Conference*, 2006.

[9] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, 2009.

[10] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, 2012.

[11] D. Kifer. Attacks on privacy and deFinetti's theorem. In *SIGMOD*, 2009.

[12] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, 2010.

[13] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets Practice on the Map. *ICDE*, 2008.

[14] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *KDD*, 2009.

[15] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially Private Data Release for Data Mining. *KDD*, 2011.

[16] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler. GUPT: privacy preserving data analysis made easy. In *SIGMOD*, 2012.

[17] S. Ruggles, J. Alexander, K. Genadek, R. Goeken, M. Schroeder, and M. Sobek. Integrated public use microdata series: Version 5.0. *Minneapolis, MN: Minnesota Population Center*, 2010.

[18] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *Internet Measurement Conference*, 2011.

[19] A. Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *STOC*, 2011.

[20] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *VLDB*, 2007.

[21] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.

[22] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *SDM Workshop*, 2010.

[23] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu. Differentially private histogram publication. In *ICDE*, 2012.

[24] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Accurate and efficient private release of datacubes and contingency tables. In *VLDB*, 2012.