DataSynthesizer: Privacy-Preserving Synthetic Datasets

Haoyue Ping
Drexel University, USA
hp354@drexel.edu

Julia Stoyanovich*
Drexel University, USA
stoyanovich@drexel.edu

Bill Howe†
University of Washington, USA
billhowe@cs.washington.edu

ABSTRACT

To facilitate collaboration over sensitive data, we present DataSynthesizer, a tool that takes a sensitive dataset as input and generates a structurally and statistically similar synthetic dataset with strong privacy guarantees. The data owners need not release their data, while potential collaborators can begin developing models and methods with some confidence that their results will work similarly on the real dataset. The distinguishing feature of DataSynthesizer is its usability — the data owner does not have to specify any parameters to start generating and sharing data safely and effectively.

DataSynthesizer consists of three high-level modules — DataDescriber, DataGenerator and ModelInspector. The first, DataDescriber, investigates the data types, correlations and distributions of the attributes in the private dataset, and produces a data summary, adding noise to the distributions to preserve privacy. DataGenerator samples from the summary computed by DataDescriber and outputs synthetic data. ModelInspector shows an intuitive description of the data summary that was computed by DataDescriber, allowing the data owner to evaluate the accuracy of the summarization process and adjust any parameters, if desired.

We describe DataSynthesizer and illustrate its use in an urban science context, where sharing sensitive, legally encumbered data between agencies and with outside collaborators is reported as the primary obstacle to data-driven governance.

The code implementing all parts of this work is publicly available at https://github.com/DataResponsibly/DataSynthesizer.

CCS CONCEPTS

• **Security and privacy → Data anonymization and sanitization**; **Privacy protections**; **Usability in security and privacy**;

KEYWORDS

Data Sharing; Synthetic Data; Differential Privacy

*This work was supported in part by NSF Grants No. 1464327 and 1539856, and BSF Grant No. 2014391.
†This work was supported by the University of Washington Information School, Microsoft, the Gordon and Betty Moore Foundation (Award #2013-10-29) and the Alfred P. Sloan Foundation (Award #3835) through the Data Science Environments program.

SSDBM '17, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3085504.3091117

ACM Reference format:
Haoyue Ping, Julia Stoyanovich, and Bill Howe. 2017. DataSynthesizer: Privacy-Preserving Synthetic Datasets. In *Proceedings of SSDBM '17, Chicago, IL, USA, June 27-29, 2017,* 5 pages.
DOI: http://dx.doi.org/10.1145/3085504.3091117

1 INTRODUCTION

Collaborative projects in the social and health sciences increasingly require sharing sensitive, privacy-encumbered data. Social scientists, government agencies, health workers, and non-profits are eager to collaborate with data scientists, but formal data sharing agreements are too slow and expensive to create in ad hoc situations — our colleagues report that 18 months is a typical timeframe to establish such agreements! As a result, many promising collaborations can fail before they even begin. Data scientists require access to the data before they can understand the problem or even determine whether they can help. But data owners cannot share data without significant legal protections in place. Beyond legal concerns, there is a general reluctance to share sensitive data with non-experts before they have "proven themselves," since they do not understand the context in which the data was collected and may be distracted by spurious results.

To bootstrap these collaborations without incurring the cost of formal data sharing agreements, we saw a need to generate datasets that are *structurally and statistically similar* to the real data but that are 1) obviously synthetic to put the data owners at ease, and 2) offer strong privacy guarantees to prevent adversaries from extracting any sensitive information. These two requirements are not redundant: strong privacy guarantees are not always sufficient to convince data owners to release data, and even seemingly random datasets may not prevent subtle privacy attacks. With this approach, data scientists can begin to develop models and methods with synthetic data, but maintain some degree of confidence that their work will remain relevant when applied to the real data once proper data sharing agreements are in place.

We propose a tool named DataSynthesizer to address this problem. Assume that the private dataset contains one table with m attributes and n tuples, and that the values in each attribute are homogeneous, that is, they are all of the same data type. We are interested in producing a synthetic dataset such that summary statistics of all numerical, categorical, string, and datetime attributes are similar to the private dataset. What statistics we preserve depends on the data type, as we will discuss in Section 3.

DataSynthesizer infers the domain of each attribute and derives a description of the distribution of attribute values in the private dataset. This information is saved in a dataset description file, to which we refer as data summary. Then DataSynthesizer is able to generate synthetic datasets of arbitrary size by sampling from the probabilistic model in the dataset description file.

DataSynthesizer can operate in one of three modes: In *correlated attribute mode*, we learn a differentially private Bayesian network capturing the correlation structure between attributes, then draw samples from this model to construct the result dataset. In cases where the correlated attribute mode is too computationally expensive or when there is insufficient data to derive a reasonable model, one can use *independent attribute mode*. In this mode, a histogram is derived for each attribute, noise is added to the histogram to achieve differential privacy, and then samples are drawn for each attribute. Finally, for cases of extremely sensitive data, one can use *random* mode that simply generates type-consistent random values for each attribute.

DataSynthesizer uses principled methods to infer attribute data types, learn statistical properties of the attributes, and protect privacy. While the specific techniques we employ are standard and have been used elsewhere (see Section 4 for a description of relevant work), the primary contribution of our system is in its usability. The system supports three intuitive modes of operation, and requires minimal input from the user.

2 DEMONSTRATION SCENARIO

We now briefly describe the main components of the DataSynthesizer system, and then explain the demonstration scenarios. The high-level architecture of the system is presented in Figure 1. The system has three modules — DataDescriber, DataGenerator and ModelInspector. Each component of the system will be explained in detail in Section 3. A data owner interacts with DataSynthesizer through a Web-based UI that serves as a wrapper for 1) invoking the DataDescriber library to compute a data summary, 2) generating a synthetic dataset from the summary using DataGenerator, and 3) inspecting and comparing datasets and data summaries with ModelInspector.

DataDescriber can be invoked on any CSV file. In particular, we can invoke it on both the input file and the output file, and compare the resulting summaries. Based on this comparison, we will convey an important point to the demonstration attendees: While the input and output datasets themselves are clearly very different, the statistical descriptions of the datasets are very similar. What sort of a comparison is drawn between the input and the output depends on the mode of operation of DataSynthesizer. When the tool in invoked in correlated attribute mode, a comparison of the learned Bayesian networks and of the pair-wise attribute correlations is shown to the user. In independent attribute mode, per-attribute histograms are presented, along with the ranges of attribute values.

During the demonstration, we will showcase the functionality of DataSynthesizer on a variety of datasets. We will prepare several interesting datasets for the interactive session, including an urban homelessness dataset from the University of Washington Data Science Institute (where the relevant tasks include making targeted service recommendations), the criminal sentencing dataset from a ProPublica investigation [1] (where the relevant tasks include analyzing the bias of recidivism models), and several datasets from the UCI Machine Learning Repository [6]. During the demonstration we will encourage users to download additional datasets from portals such as data.seattle.gov and data.ny.gov and to run the tool.

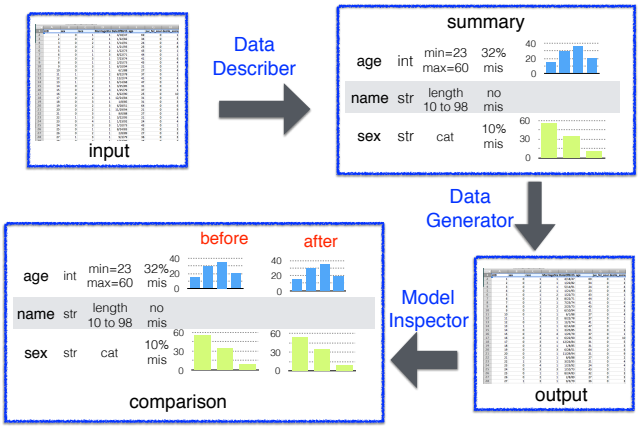

Figure 1: The DataSynthesizer system architecture.

Attendees will play the role of a data owner who is preparing a synthetic dataset for release to a set of new collaborators. After reviewing the raw dataset, they will generate a summary model and inspect the results, verifying that the results are sensible. Then, the attendees will generate a synthetic dataset from this model and inspect individual records, observing that the records are visibly "scrambled". Finally, they will generate a new model from the synthetic dataset and observe that the statistical properties are intact. At this point, we will explain the privacy guarantees that are enforced, and will illustrate them using the before-and-after visualizations.

3 SYSTEM OVERVIEW

DataSynthesizer is an end-to-end system that takes a private dataset as input and generates synthetic datasets. The system is implemented in Python 3. It assumes that the private dataset is presented in CSV format.

The input dataset is first processed by the DataDescriber module. The domains and the estimates of distributions of the attributes are inferred and saved in a dataset description file. For each attribute identified as categorical, DataDescriber computes the frequency distribution of values, represented as a bar chart. DataGenerator then samples from this distribution when deriving the synthetic dataset. For non-categorical numerical and datetime attributes, DataDescriber derives an equi-width histogram to represent the distribution. DataGenerator draws samples from this histogram during data generation using uniform sampling. For non-categorical string attributes, their minimum and maximum lengths are recorded. DataDescriber generates random strings within the length range during data generation stage.

We now describe each of the modules of DataSynthesizer in turn.

3.1 DataDescriber

3.1.1 Inferring data types and domains. The domain of an attribute is the set of its legal values. The data type is an important ingredient of the attribute domain. DataSynthesizer supports four data types. The system allows users to explicitly specify attribute data types. If an attribute data type is not specified by the user,

Table 1: Data types supported by DataSynthesizer

Data Type	Example
integer	id, age, ...
float	score, rating, ...
string	name, gender, ...
datetime	birthday, event time, ...

it is inferred by the DataDescriber. For each attribute, DataDescriber first detects whether it is numerical, and if so — whether it is an *integer* or a *float*. If the attribute is non-numerical, DataDescriber attempts to parse it as *datetime*. Any attribute that is neither numerical nor *datetime* is considered a *string*.

The data type of an attribute restricts its domain, which may be limited further if the attribute is categorical, where only specific values are legal. For example, in a dataset of students, the attribute *degree* may only take on values *BS*, *BA*, *MS*, or *PhD*. The domain of *degree* is {*BS*, *BA*, *MS*, *PhD*}. Note that *integer*, *float* and *datetime* attributes can also be categorical. In general, an attribute is considered to be categorical if a limited number of distinct values for that attribute is observed in the input dataset. DataDescriber has a parameter *categorical threshold* that defaults to 10 and represents the maximum number of distinct values for a categorical attribute.

The *categorical threshold*, like any threshold, may be challenging, or even impossible, to set in a way that reflects user preferences for all attributes in the input. Some attributes may appear categorical, but the user may prefer to treat them as numerical instead. For example, age of elementary school children may take on only a handful of distinct values but the user may nonetheless wish to generate data for this attribute from a continuous range. The opposite situation may also arise — an attribute may take on 200 or so distinct values, as is the case with country names, and so would not be considered categorical under any reasonable threshold. Still, the user may prefer to treat this attribute as categorical, so that a valid country name, rather than a random string, is generated for this attribute in the synthesized dataset. For this reason, DataSynthesizer allows users to specify a data type, and state whether an attribute is categorical, over-riding defaults on a per-attribute basis.

Note that the actual datatype of a categorical attribute is immaterial in terms of the statistical properties and the privacy guarantees in synthetic data generation. For example, an attribute such as *sex* may be encoded as M/F, as 0/1 or using a Boolean flag (e.g., True for male and False for female). In all cases, the tool will compute the frequencies of each attribute value in the input, and will subsequently sample values from the resulting data summary.

There might be missing values in the input dataset, and these are important to represent in the summary. DataDescriber calculates the missing rate for each attribute — the number of observed missing values divided by n, the size of the dataset.

3.1.2 Differential privacy. Differential privacy is a family of techniques that guarantee that the output of an algorithm is statistically indistinguishable on a pair of *neighboring* databases; that is, a pair of databases that differ by only one tuple. This concept is formalized with the following definition.

Algorithm 1 GreedyBayes(D, A, k)

Require: Dataset D, set of attributes A, maximum number of parents k

Initialize $\mathcal{N} = \emptyset$ and $V = \emptyset$.
Randomly select an attribute X_1 from A.
Add $(X_1, \emptyset)$ to $\mathcal{N}$; add X_1 to V.
for $i = 2, .., |A|$ **do**
  Initialize $\Omega = \emptyset$
  $p = min(k, |V|)$
  for each $X \in A \backslash V$ and each $\Pi \in \binom{V}{p}$ **do**
    Add (X, Π) to Ω
  end for
  Compute mutual information based on D for all pairs in Ω.
  Select (X_i, Π_i) from Ω with maximal mutual information.
  Add (X_i, Π_i) to $\mathcal{N}$.
end for
return $\mathcal{N}$

(ϵ-Differential Privacy [3]) Let ϵ be a positive number. Let $\mathcal{A}$ be a randomized algorithm taking a dataset as input. Let D be a dataset. For any D' that differs from D on at most one tuple, for any legal output O of $\mathcal{A}$, $Pr(\mathcal{A}(D) = O) \leq e^{\epsilon} \times Pr(\mathcal{A}(D') = O)$.

When ϵ approaches 0, $Pr(\mathcal{A}(D) = O) = Pr(\mathcal{A}(D') = O)$. That is, the presence or absence of a single individual in the input to the algorithm will be undetectable when one looks at the output.

3.1.3 Independent attribute mode. When invoked in independent attribute mode, DataDescriber performs frequency-based estimation of the unconditioned probability distributions of the attributes. The distribution is captured by bar charts for categorical attributes, and by histograms for numerical attributes. Precision of the histograms can be refined by a parameter named *histogram size*, which represents the number of bins and is set to 20 by default.

DataDescriber implements a differentially private mechanism, adding controlled noise into the learned distributions. The noise is from a Laplace distribution with location 0 and scale $\frac{1}{n\epsilon}$, where n is the size of the input, denoted $Lap(\frac{1}{n\epsilon})$, setting $\epsilon = 0.1$ by default. DataDescriber also adds Laplace noise to the missing rate. When Laplace noise is added to histogram frequencies, the value may become negative. In that case the value is reset to 0 [8].

3.1.4 Correlated attribute mode. Attribute values are often correlated, e.g., *age* and *income* of a person. When invoked in correlated attribute mode, DataDescriber uses the GreedyBayes algorithm to construct Bayesian networks (BN) to model correlated attributes [8].

Algorithm 1 constructs a BN $\mathcal{N}$ from input dataset D, attributes A, and the maximum number of BN node parents k, which defaults to 4. In this algorithm, V is the set of visited attributes, and Π is a subset of V that will become parents of node X if added to $\mathcal{N}$. Which attributes Π are selected as parents of X is determined greedily, by maximizing mutual information (X, Π). Algorithm 1 learns the structure of the BN with privacy guarantees when the mutual information computation is differentially private [8].

The Bayesian networks constructed in Algorithm 1 gives the sampling order for generating attribute values. The distribution from which a dependent attribute is sampled is called a conditioned

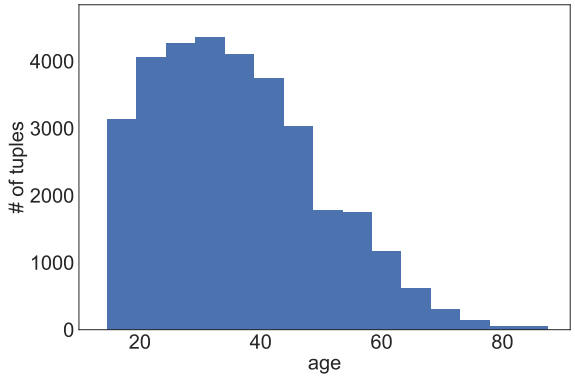

Figure 2: Histogram on *age*: Adult Income [6].

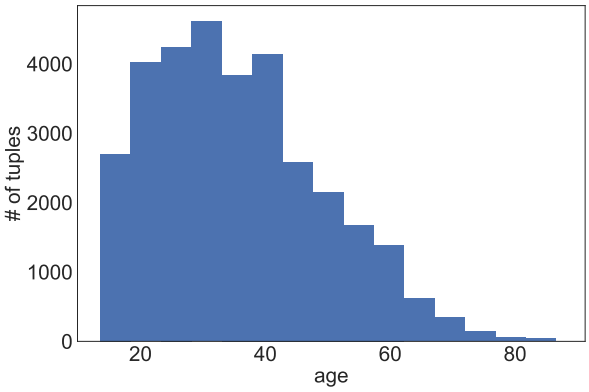

Figure 3: Histogram on *age*: synthetic.

Figure 4: Pair-wise correlations: Adult Income [6].

Figure 5: Pair-wise correlations: synthetic.

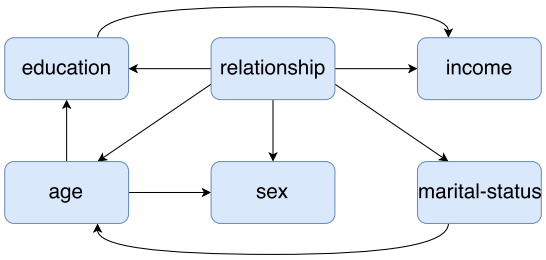

Figure 6: Bayesian network: Adult Income [6].

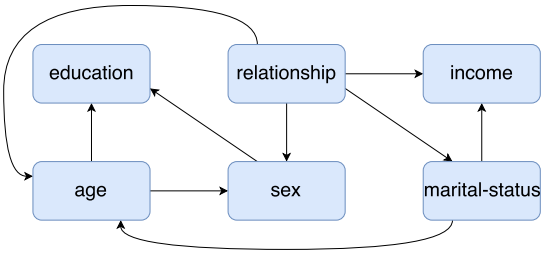

Figure 7: Bayesian network: synthetic.

distribution. When constructing noisy conditioned distributions, $Lap(\frac{4(d-k)}{n \cdot \epsilon})$ is injected to preserve privacy. Here, d is the number of attributes, k is the maximum number of parents of a BN node, and n is the number of tuples in the input dataset. We construct conditional distributions according to Algorithm 1 of [8].

The parents of a dependent attribute can be categorical or numerical, whose distributions are modeled by bar charts and histograms, respectively. The conditions for this dependent attribute are the legal values of the categorical parents and the intervals of the numerical parents. Here, the intervals are formed in the same way as the unconditioned distributions of the parent attributes. For example, the *age* attribute has intervals {[10, 20),[20, 30),[30, 40)} in its unconditioned distribution. Assume *education* only depends on *age*. Its conditioned distributions will be under the same intervals, i.e., age $\in [10, 20)$, age $\in [20, 30)$ and age $\in [30, 40)$ respectively.

Algorithm 2 DataGenerator($n, \mathcal{M}, \mathcal{S}, A_U, s$)

Require: number of tuples n to generate, mode $\mathcal{M}$, dataset description $\mathcal{S}$, uniform attributes A_U, seed s

```
Set seed = s for pseudo-random number generator.
if M is independent attribute mode then
    Read all attributes A from S.
    for X ∈ A do
        if X ∈ A_U then
            Read the domain of X from S.
            Sample n values uniformly from its domain.
        else
            Read the distribution of X from S.
            Sample n values from its distribution.
        end if
    end for
else if M is correlated attribute mode then
    Read Bayesian network N from S.
    Sample root attribute from an unconditional distribution.
    Sample remaining attributes from conditional distributions.
end if
return Sampled dataset
```

3.2 DataGenerator

Given a dataset description file generated by DataDescriber, DataGenerator samples synthetic data from the distributions in this file. The size of the output dataset is specified by the user, and defaults to n, the size of the input dataset.

Algorithm 2 describes the data generation process. When invoked in random mode, DataGenerator generates type-consistent random values for each attribute. When invoked in independent attribute mode, DataGenerator draws samples from bar charts or histograms using uniform sampling. Finally, when invoked in correlated attribute mode, DataDescriber samples attribute values in appropriate order from the Bayesian network.

An important property of differential privacy is that its performance degrades with repeated queries. To prevent leaking private information through adversaries repeatedly sending data generation requests, the system administrator can assign a unique random seed for each person who requires a synthetic dataset. To support this, DataGenerator provides per-user seed functionality.

3.3 Model Inspector

ModelInspector provides several built-in functions to inspect the similarity between the private input dataset and the output synthetic dataset. The data owner can quickly test whether the tuples in the synthetic dataset are detectable by inspecting and comparing the first 5 and last 5 tuples in both datasets.

The synthetic dataset should have similar distributions of attribute values as the input dataset. In independent attribute mode, ModelInspector allows users to visually compare attribute distributions in the form of bar charts or histograms. Figures 2 and 3 present such summaries for the attribute *age* from the Adult Income dataset [6]. The system also computes the correlation coefficient of an appropriate kind, depending on attribute type, and the KL-divergence value, which measure how much the "after" probability distribution diverges from the "before" probability distribution for a given attribute. In correlated attribute mode, ModelInspector presents "before" and "after" pairwise mutual information matrices (Figures 4 and 5) and describes Bayesian networks (such as those shown graphically in Figures 6 and 7), enabling an at-a-glance comparison of the statistical properties of the datasets.

4 RELATED WORK

In our work on DataSynthesizer we leverage recent advances in practical differential privacy [4] and privacy-preserving generation of synthetic datasets [7, 8]. In particular, we make use of the privacy-preserving learning of the structure and conditional probabilities of a Bayesian network in PrivBayes [8], and are inspired in our implementation by the work on DPBench [4].

Data sharing systems, including SQLShare [5] and DataHub [2], aim to facilitate collaborative data analysis, but do not incorporate privacy preserving features or purport to manage sensitive data. We see these systems efforts as a potential delivery vector for DataSynthesizer capabilities.

5 TAKE-AWAY MESSAGES

We have presented a demonstration of DataSynthesizer, a privacy-preserving synthetic data generator designed to facilitate collaborations between domain-expert data owners and external data scientists. The cost of establishing formal data sharing agreements limits the impact of these ad hoc collaborations in government, social sciences, health, or other areas where data is heavily encumbered by privacy rules. Given a dataset, DataSynthesizer can derive a structurally and statistically similar dataset, at a configurable level of statistical fidelity, while ensuring strong privacy guarantees. DataSynthesizer is designed with usability in mind. The system supports three intuitive modes of operation, and requires minimal input from the user.

We see DataSynthesizer being used in a variety of application contexts, both as a stand-alone library and as a component of more comprehensive data sharing platforms. As part of ongoing work, we are studying how best to deliver these features to data owners, and determining how additional requirements can be met. DataSynthesizer is open source, and is available for download at https://github.com/DataResponsibly/DataSynthesizer.

REFERENCES

[1] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine Bias. *ProPublica* (23 May 2016). https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[2] Anant P. Bhardwaj and others. 2015. DataHub: Collaborative Data Science & Dataset Version Management at Scale. In *CIDR*.

[3] Cynthia Dwork and others. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*.

[4] Michael Hay and others. 2016. Principled Evaluation of Differentially Private Algorithms using DPBench. In *SIGMOD*.

[5] Shrainik Jain and others. 2016. SQLShare: Results from a Multi-Year SQL-as-a-Service Experiment. In *SIGMOD*. ACM, New York, NY, USA.

[6] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml

[7] Wentian Lu and others. 2014. Generating private synthetic databases for untrusted system evaluation. In *ICDE*.

[8] Jun Zhang and others. 2014. PrivBayes: private data release via Bayesian networks. In *SIGMOD*.