Harvard Privacy Tools Project

Data Processing Notes

Christian Covington

April 1, 2020

1 Overview

This document contains notes on our choices regarding "data processing", which I am using to stand in for all things related to data bookkeeping (data bounds, imputation, sample size calculations, etc.).

2 Clamping

Many operations in the library require the definition of a set of possible data values we would like to use for said operation. For numeric variables, this typically takes the form of a closed interval $[min, max] \in \mathbb{R}$ or $\mathbb{Z}$. For categorical variables, this is a discrete set of elements. In our system, we have slightly different notions of clamping for numeric and categorical variables. We consider $f64$ and $i64$ to be allowable numeric types, and $i64, bool$, and $String$ to be allowable categorical types.

Our method for clamping numeric types should be familiar; it accepts data, a min, and a max, and maps data elements outside of the interval $[min, max]$ to the nearest point in the interval. Null values (represented as NAN for the $f64$ type and which do not exist for the $i64$ type) are preserved by numeric clamping.

Our categorical clamping accepts data, a set of feasible values, and a null value. Data elements that are outside the set of feasible values are mapped to the null value. Null values are preserved by the categorical clamping.

3 Imputation

Much like clamping, we have slightly different notions of imputation for numeric and categorical variables. In this case, we consider only $f64$ to be an allowable numeric type, while

*i*64, *bool*, and *String* are allowable categorical types.

Numeric imputation is parameterized by min, max, and a supported probability distribution (e.g. Gaussian or Uniform) with appropriate arguments. *NAN* values are replaced with a draw from the provided distribution.

Categorical imputation is parameterized by categories, probabilities, and a null value. Each data element equal to the null value is replaced with a draw from the set of categories, according to the probabilities provided.

4 Public vs. Private n

We want our library to support cases both in which the analyst does and does not have access to the number of records in the data. This is an under-explored problem, so we provide our strategy for dealing with it as well as some potential alternatives.

4.1 Our strategy

We require that, when asking for a statistic, the analyst must choose an $\hat{n}$ – an estimate of the true sample size n. If $\hat{n} = n$, then the analysis proceeds on the true data, x. If $\hat{n} \neq n$, the we either subsample or create synthetic data to create a data set, $\tilde{x}$, of size $\hat{n}$ to be used for the analysis.

4.1.1 Choosing $\hat{n}$

We offer two ways to choose an $\hat{n}$; the analyst make their own guess about n, or they can consume some privacy budget to estimate n. In the case of a user-provided $\hat{n}$, this could be because n is actually public knowledge (in which case the user's estimate should be correct), or it could be a guess by the user. Whether $\hat{n}$ is provided by the user or estimated, this $\hat{n}$ will be used throughout the entire analysis.

We chose this strategy for a few reasons. First, we want the library to be general and this strategy seemed to meet that criterion. Second, we did not feel comfortable having different processes for public vs. private n, e.g. one in which the library automatically finds the sample size from public metadata and treats it as ground truth if the metadata states that $n \neq \textit{private}$. Because public metadata are not typically assumed to be stored securely, it is possible that a metadata file could be changed nefariously, for example from $n = \textit{private}$ to $n = 1,000$ (and thus $n \neq \textit{private}$). In such a case, we would not want the library to accept sample sizes from public metadata without suspicion; if it did so, then it would likely throw errors if the actual size of the data were not n, thus revealing to the adversary that the actual size of the data $\neq 1,000$. By not distinguishing between public and private n and treating any user-input as a guess, we are making our system robust to metadata attacks. This does come with the downside that we have no notion of verifying n within the library, this verification must come from users' trust in the veracity of the public metadata.

4.1.2 Making data consistent with $\hat{n}$

If we need to create synthetic data, we generate $\hat{n} - n$ new elements using the user-provided data bounds specified for the statistic in question. This could be done via any imputation method supported by the library. Because these imputed values are within the data bounds

and independent of the actual data, our function sensitivity calculations remain correct.

If we need to subsample, we simply subsample without replacement to the desired size. Note that we cannot take advantage of any privacy amplification via subsampling results, as this would leak information about the relationship between $\hat{n}$ and n.

4.2 Accuracy/Error

Our accuracy and error calculations in the library always assume that $\tilde{x}$ is the true data in question. That is, the additional error induced by subsampling or imputing extra values is not considered.

4.2.1 Maintaining Subset Consistency

Consider the case where $\hat{n} < n$ such that the library will subsample the data before calculating the statistic of interest. There are many cases in which the library should ensure subsampling across multiple steps with a consistent set of indices. For example, say the analyst wants to add two columns together or calculate a covariance – these operations become meaningless without a guarantee that the columns in question retain their relationship to each other when subset.