Differentially Private Linear Regression

Christian Baehr

August 5, 2017

Abstract

Your abstract.

1 Introduction

My research involved testing and implementing tools into the Harvard Privacy Tools Project PSI tool. PSI refers to "A Private data Sharing Interface" and is being developed as a means for researchers to generate and access summary statistics while maintaining strong privacy guarantees for individuals in the data. The PSI tool contains two main components: a front-end user interface for researchers to interact with and request differentially private statistics, and a back-end containing differential privacy algorithms for computing various statistics in a differentially private way. The back end of PSI consists primarily of a developing R package. Most of the work I did this summer involved development of the package. My primary task was to prepare the package to be released to the public, and a secondary task was to test and implement methods for performing linear regression analysis on data in a differentially private way.

2 R Package

2.1 What is an R package?

R is an open-source statistical programming language. It is commonly used in the social sciences. Many researchers and methodologists will share their methods with the R community in the form of packages. An R package contains functions, compiled code, and documentation, and often includes additional features, such as examples, vignettes, and test data. The back end of PSI is written in R and will be released as an R package to allow researchers the option of interacting with PSI algorithms without having to go through the user interface.

2.2 Package Development

To release a package for use by the R community, the package developer must ensure the package passes a number of checks designated by the Comprehensive R Archive Network, or CRAN. These checks are designed to ensure package code runs properly, code is appropriately documented, package metadata is accurate, among other things.

My role in developing the package included several components. The first was to make sure the package passed the CRAN checks. I also completed any missing documentation, ensuring an explanation existed of each function in the package as well as a brief description of each input and output in the function. I added informative examples for functions users were likely to interact with, demonstrating how users should specify function inputs. To clearly demonstrate to users how functions in the package interact with each other, I created a vignette that explains in greater detail how to use the `mean.release()` function. This vignette should assist users in understanding not only this function, but also the functions for releasing other statistics such as histograms, covariance matrices, and regression coefficients. The vignette is included in Appendix E. The package is, in its current state, prepared to be sent to CRAN.

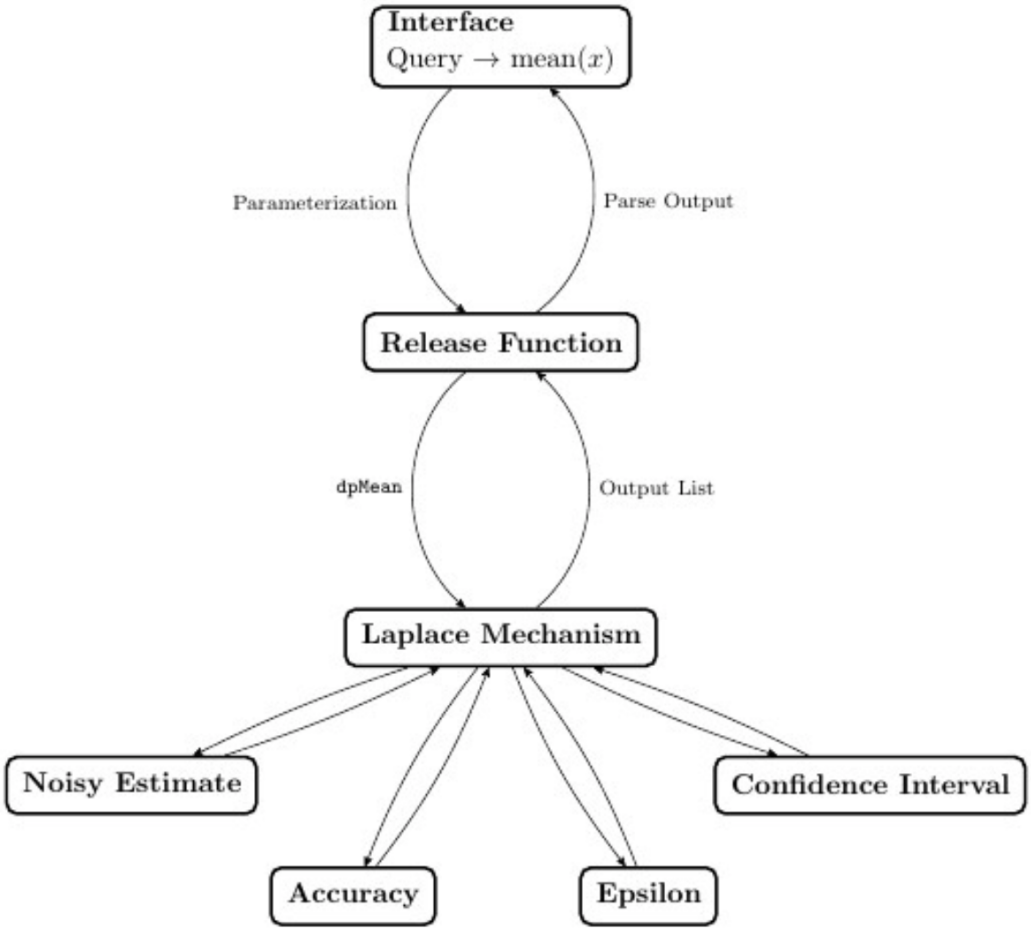


Figure 1: The structure of the `mean.release()` function

2.3 Package Structure

The PSI R package is structured in a modular fashion that is optimal for facilitating contributions by users and also assures private data will be touched as little as possible. Figure 1 displays the structure of the `mean.release()` function. If the user is interacting directly with the R package, they will skip the top step in the diagram and will instead input their parameters directly into the `mean.release()` function in R. It is noteworthy the user must manually specify the number of observations, variable type, and range of the data of interest; this ensures the package will touch the data as little as possible, and thus decreases the possibility of an unauthorized release of information. Additional parameter specification includes the desired epsilon value and post-processing steps the user would like to perform on the released statistic.

Once the user executes the `mean.release()` function, the release function first ensures the input specification is acceptable, then submits the request to the `mechanism.laplace()` function. This mechanism extracts the true value of the statistic from `dp.mean()` and adds noise from the LaPlace distribution to make the statistic differentially private. Once the differentially private algorithm is run on the statistic, post-processing can be done on the statistic without releasing any additional private information. Post-processing steps on a differentially private mean can include estimation of confidence interval, an accuracy guarantee for the mean estimate, and, if the variable is logical, a median of the data. After these steps are completed, the user is returned a list containing the name of the statistic, variable type, number of observations, sensitivity of the data, the epsilon value, the differentially private mean, and whatever post-processing steps are requested.

3 Differentially Private Linear Regression

3.1 Objective

A portion of my work with the project was dedicated to finding optimal strategies for obtaining differentially private regression coefficients. There are existing methods for obtaining private coefficient estimates, and most of these do ensure a strong privacy guarantee for the user; however, many of these mechanisms require so much noise be added to the computation that estimates are inaccurate and offer little to no utility to the user. My objective was to seek the existing methods that offered the most accurate coefficient estimates while maintaining the privacy of observations.

I considered two algorithms for predicting differentially private regression coefficients: (1) perturbation of the covariance matrix as discussed in (Dwork et al. 2014), and then, as a post-processing step, sweeping the coefficients from the differentially private matrix, and (2) perturbing the objective function, based on (Chaudhuri et al. 2011), and maximizing the perturbed objective to obtain optimal coefficient estimates.

3.2 Covariance Perturbation

One method of obtaining differentially private coefficient estimates is by first computing the true covariance matrix for the data of interest and adding noise from a differentially private algorithm to make the covariance matrix (ϵ, δ) differentially private. Because post-processing cannot harm privacy (Dwork + Roth 2014), we can, as a post-processing step, sweep coefficient estimates from this differentially private covariance matrix without revealing any additional information about individuals in the data and, therefore, without spending any more epsilon.

I considered two different noise-generating mechanisms for generating a differentially private covariance matrix. Per (Dwork et al. 2014), I used the Gaussian normal distribution as a noise generating mechanism, and I also evaluated the efficacy of the LaPlace distribution. The PSI R library had an existing mechanism to generate differentially private covariance matrices and a sweep operator for extracting coefficients, so I simply had to combine these components, input data, and run simulations. To test the performance of each noise mechanism, I generated synthetic random multivariate normal data.

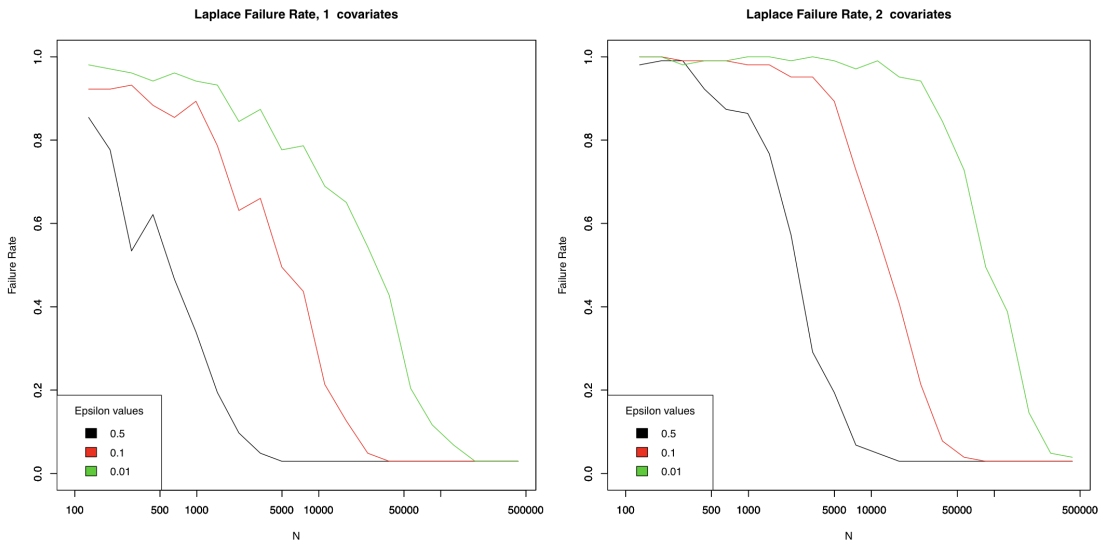


Figure 2: The failure rate of the private VCV matrix to invert with LaPlace noise

One metric of comparison between the two noise mechanisms is how closely noisy coefficient estimates adhered to the true values. Another consideration in evaluating the performance of the noise mechanisms is the rate at which the resulting matrix is unable to invert. In some cases, noise

addition to a covariance matrix may cause it to be near singular and thus non-invertible. I will refer to the rate at which the DP covariance matrix fails to invert as the "failure rate", because attempting to sweep coefficients from a non-invertible covariance matrix will result in extreme estimates.

In addition to the noise mechanism, determinants of an increased failure rate include fewer observations in the data, increased covariates in the data, and a larger epsilon parameter. As each graph in Figure 2 reflects, as N, the number of observations, increases, the failure rate falls and the noisy covariance matrix is invertible at a higher rate. Likewise, as the epsilon value grows and the privacy of the release becomes weaker, the amount of noise and the failure rate both drop. A comparative look at the graphs reveals that with a greater number of covariates, the graph will fail at a higher rate with fixed N and epsilon.

The primary measure of how well a differentially private regression algorithm is performing is how closely coefficient estimates adhere to true values. The histograms in Figure 3 display the distribution of the intercept coefficient in a linear regression with three covariates over 1000 simulations. A comparative analysis of these distributions (with a careful eye on the relative x-axis limits) reveals the LaPlace mechanism generates much more accurate coefficients than the Gaussian mechanism. All coefficient estimates from the LaPlace distribution fall nearly within the range (-0.04, 0.06) while the range of estimates from the Gaussian mechanism is much greater, roughly (-0.2, 0.3).

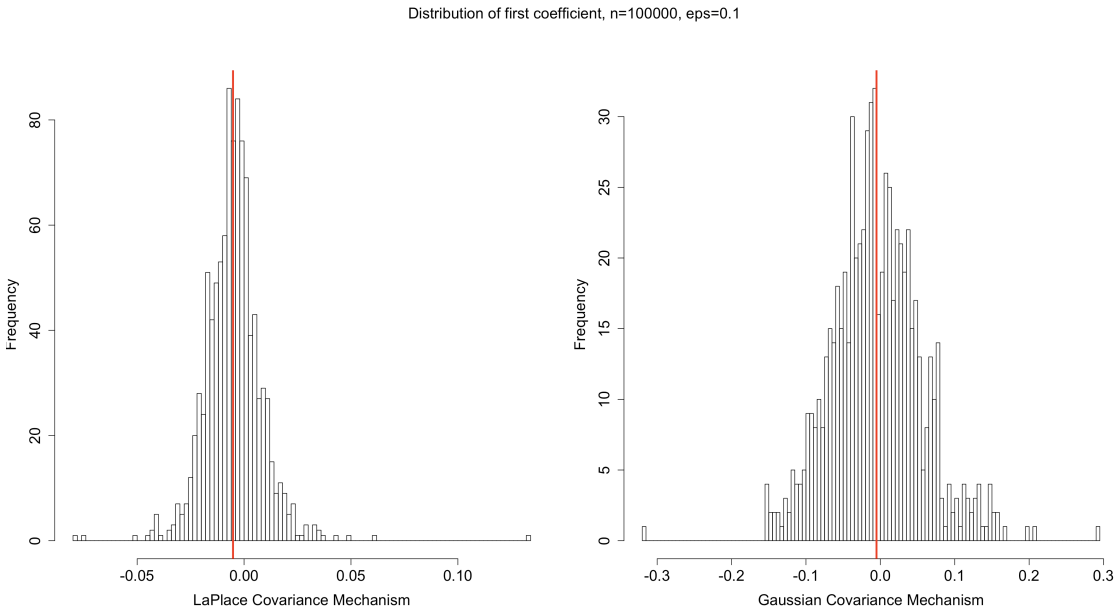


Figure 3: Distribution of the intercept coefficient (true value denoted in red)

Additional histograms comparing the second and third coefficient estimates can be found in Appendix A and B, and they reveal similar conclusions.

3.3 Objective Perturbation

The second algorithm I tested for obtaining regression coefficients is the objective perturbation method. It is so named because, in this method, differential privacy is achieved by adding noise to the log-likelihood function, also known as the objective function. Regression coefficients are obtained by maximizing the log-likelihood function of the regression model being used. For least squares regression, the coefficients that maximize the objective function are those values which minimize the sum of squared residuals. This is how the `lm()` function in R evaluates which coefficients to return.

If n is the number of observations in the data, and σ^2 and β_i are the variance and coefficients we are estimating,

$$logL = -\frac{n}{2} * \log 2\pi - \frac{n}{2} * \log \sigma^2 - \frac{1}{2 * \sigma^2} * \sum_{i}^{n} (y_i - X_i\beta_i)^2$$

is the log likelihood function we perturb to extract differentially private coefficient estimates. By adding noise from a LaPlace distribution and dividing the perturbed objective by n and maximizing, we obtain differentially private coefficient estimates. Because the variance is a parameter in the log likelihood function for least squares regression, we obtain an estimate for variance as well.

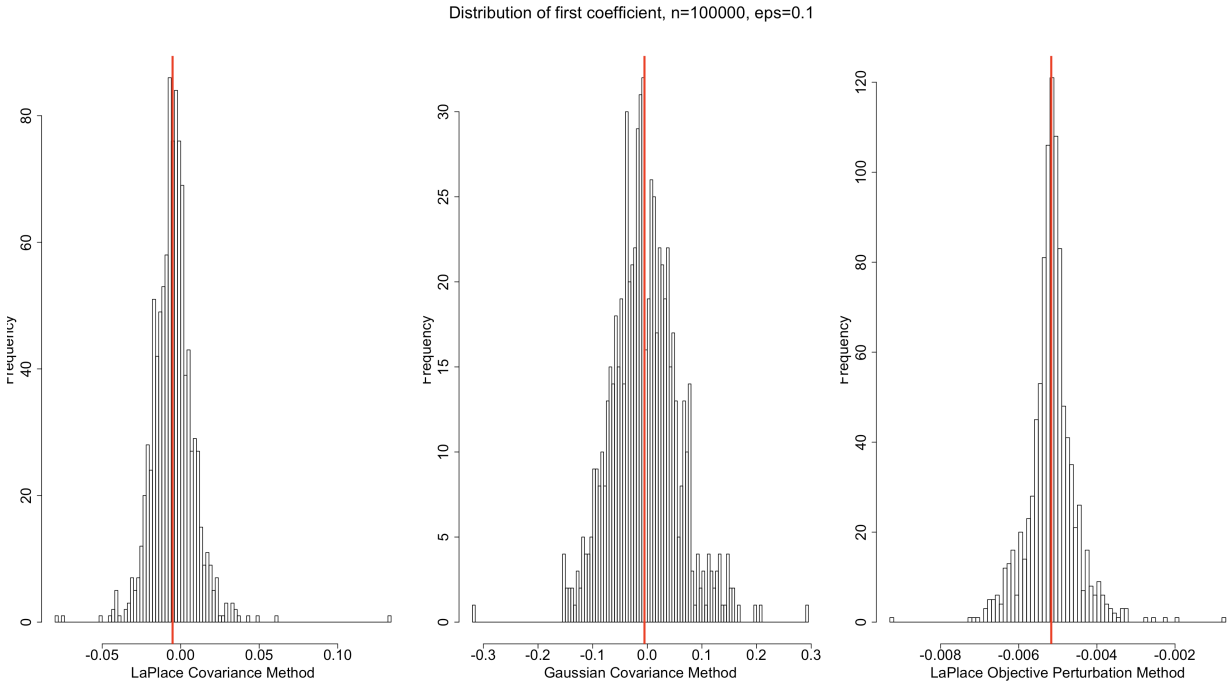


Figure 4: Distribution of the intercept coefficient (true value denoted in red)

The histograms in Figure 4 display the distributions of the intercept coefficient just as the previous figure did, but now including the Objective Perturbation distribution. Again, a quick look at the axis ranges for each plot reveals how each mechanism is performing relative to the others. The range is much greater the Gaussian covariance estimates, roughly (-0.3, 0.3). A comparison of the LaPlace methods reveals that adding LaPlace noise to the objective function returns much more accurate estimates, with a range of roughly (-0.008, -0.002), than LaPlace noise on the covariance matrix, which has a range of roughly (-0.05, 0.10). This indicates the objective perturbation method produces much more accurate coefficient estimates than either the LaPlace or Gaussian noise mechanism on a covariance matrix. See Appendix C and D for the histograms of the distributions of the second and third coefficients.

3.4 Discussion

Based on the coefficient spreads with equivalent observations, epsilon values, and number of covariates, it is apparent objective perturbation returns the most accurate coefficient estimates of the three mechanisms. It is also notable that objective perturbation does not face the issue of a near singular matrix that plagues the covariance method at lower n and higher numbers of covariates. However, a quality of sweeping coefficients from a covariance matrix is that it is post-processing, and therefore a researcher may run as many different formulas as they like on the differentially private matrix without expending any more epsilon. With this is mind, there is some utility in keeping the functionality to sweep coefficients from the covariance matrix in the package, because at high n and few covariates, researchers may still get accurate results and have more freedom to explore various relationships between variables. It is also apparent LaPlace noise should be used

when adding noise to a covariance matrix. I would, however, recommend the objective perturbation mechanism to any researcher who seeks accurate results without the luxury of high n, or who is only interested in exploring one relationship in the data.

4 References

Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. JMLR, 12:1069–1109, 2011.

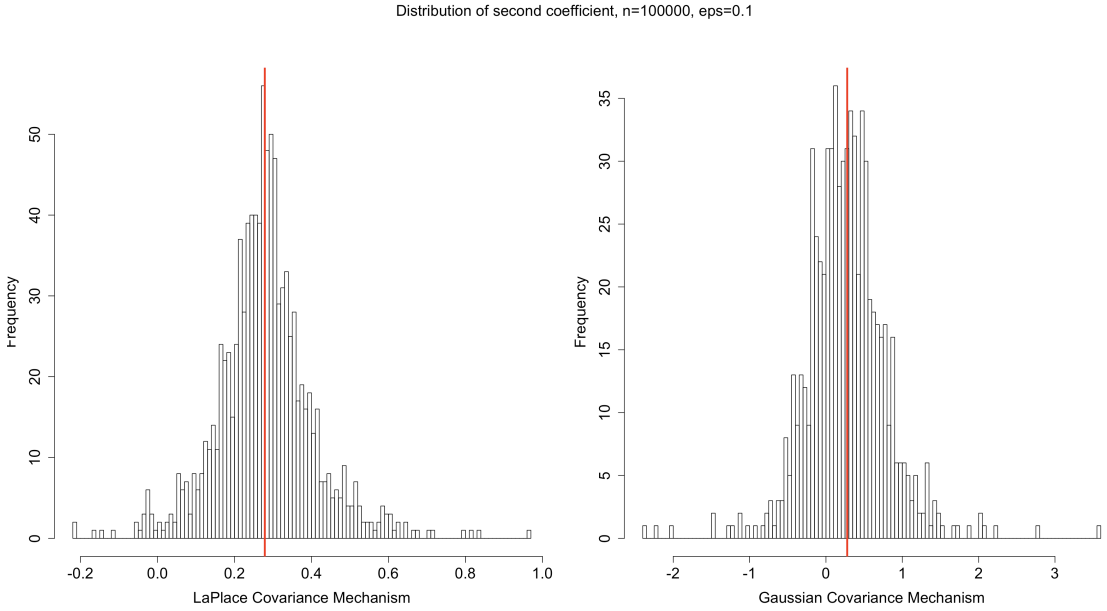
D. Kifer, A. D. Smith, and A. Thakurta. Private convex optimization for empirical risk minimization with applications to high-dimensional regression. Journal of Machine Learning Research - Proceedings Track, 23:25.1–25.40, 2012.

Dwork, C.; Talwar, K.; Thakurta, A.; and Zhang, L. 2014. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing, 11–20. ACM.

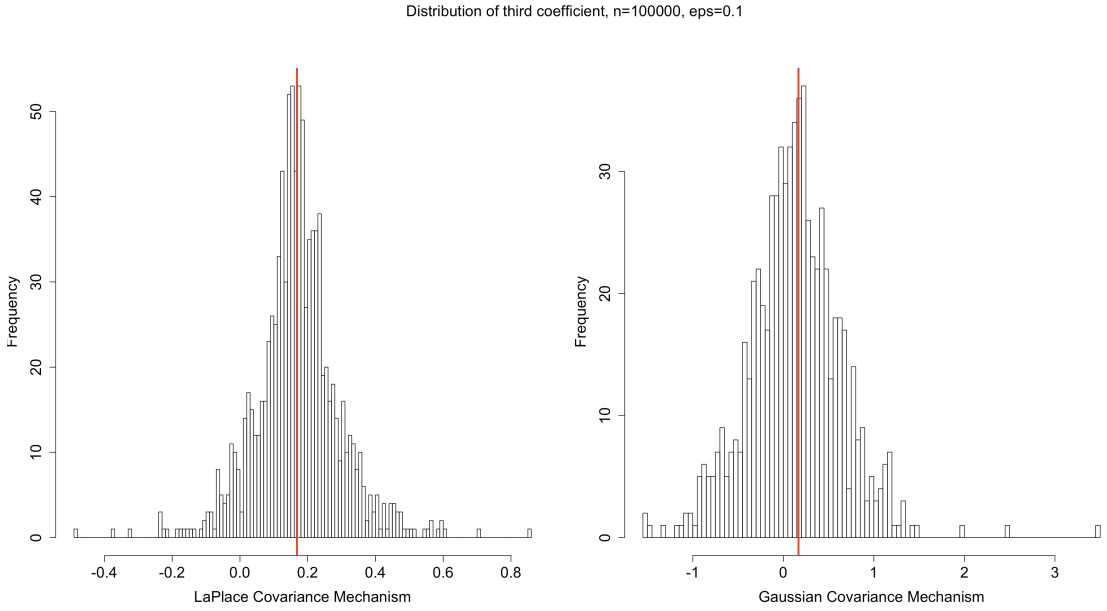
Cynthia Dwork , Aaron Roth, The Algorithmic Foundations of Differential Privacy, Foundations and Trends® in Theoretical Computer Science, v.9 n.3–4, p.211-407, August 2014[doi>10.1561/0400000042]

5 Appendix

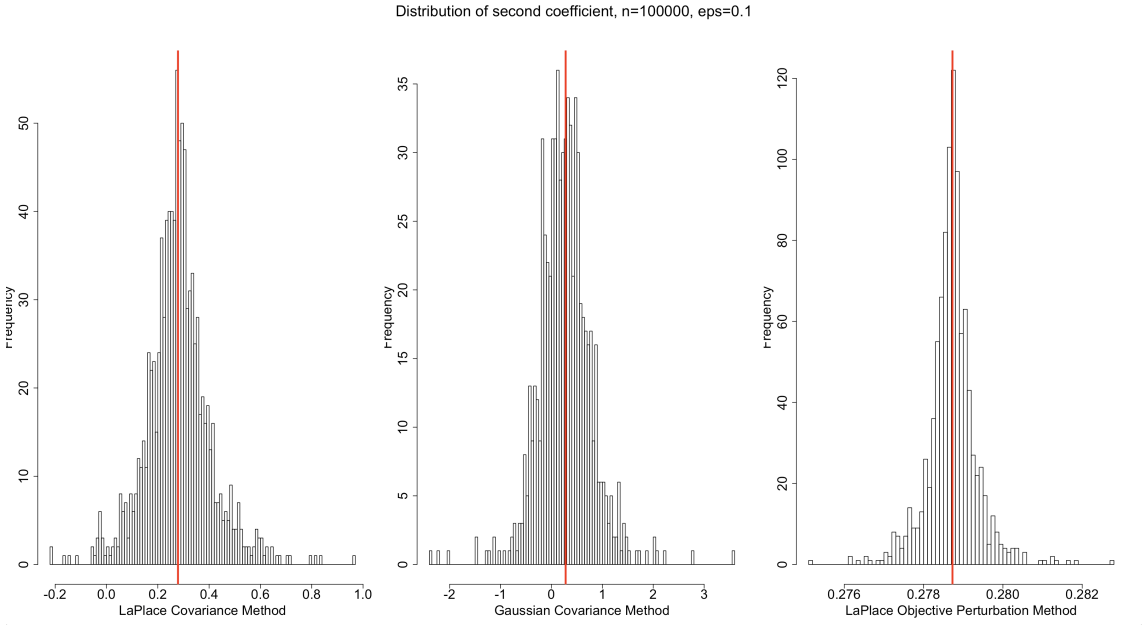
5.1 Appendix A



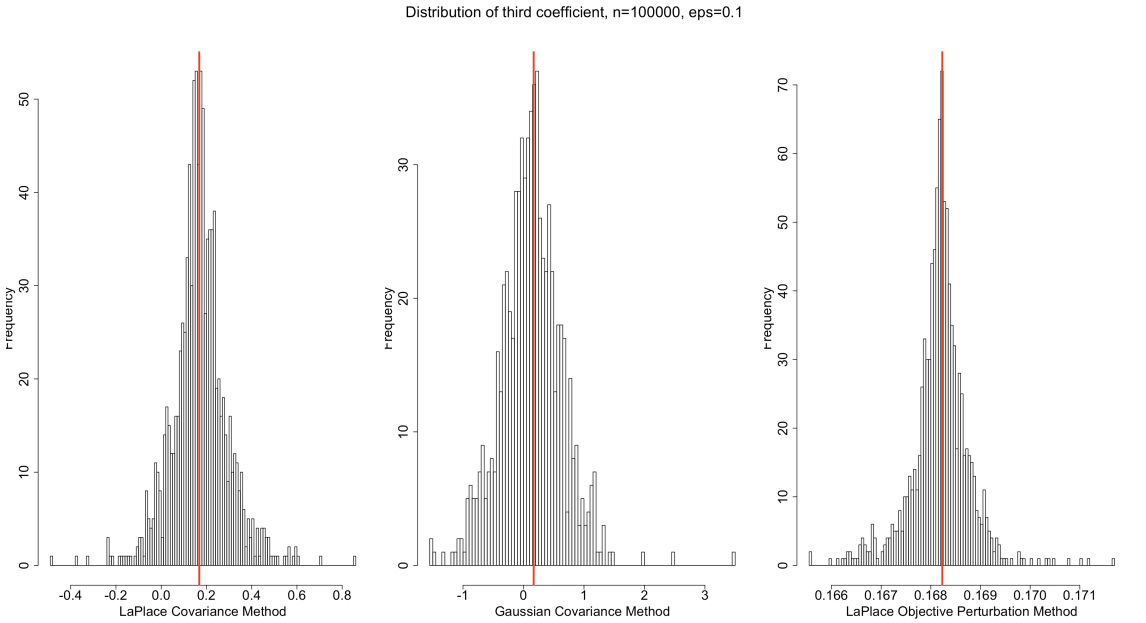
5.2 Appendix B



5.3 Appendix C



5.4 Appendix D



5.5 Appendix E

Differentially Private Mean Release

James Honaker, Thomas Brawner, and Christian Baehr

2017-08-05

Calculate Differentially Private Mean of a Vector with `mean.release`.

Differential privacy is a rigorous mathematical framework for making statistical information about private datasets available. This is done in such a way that guarantees that information about specific individuals in the dataset does not leak out. Differentially private versions of various statistics are available in this package. For example, the `mean.release()` function releases a mean of a variable, while adding a precise about of noise to guarantee `epsilon` differential privacy.

The privacy loss parameter `epsilon` is chosen by the user and represents the degree of privacy preservation guaranteed to each observation in the data when releasing information. Typically this is valued between 0 and 1; as the value gets smaller, the level of privacy protection grows. However, greater privacy protection means more noise must be added to the true mean to achieve the desired amount of privacy. Therefore, as epsilon grows smaller, the privacy protection becomes greater, but the accuracy of the statistical release becomes weaker.

Syntax

```
mydata <- c(3, 12, 20, 42, 33, 65, 70, 54, 33, 45)
private.mean <- mean.release(x = mydata, var.type = "numeric", epsilon = 0.1, n = 10, rng = c(3,
70))

mydata2 <- c(TRUE, FALSE, FALSE, TRUE, FALSE, TRUE, TRUE, TRUE)
private.mean2 <- mean.release(x = mydata2, var.type = "logical", epsilon = 0.5, n = 8, rng = c(0,
1))
```

Inputs

`mean.release()` accepts the following arguments to specify the the desired level of privacy and characteristics of the variable of interest.

- `x`: The vector of data for which the differentially private mean will be calculated. This data can be of type numeric, integer, or logical.
- `var.type`: A character vector of length one identifying the data type of `x`. This field should contain either ‘numeric’, ‘integer’, or ‘logical’.
- `epsilon`: A numeric vector of length one specifying the epsilon privacy parameter. This should be between zero and one.
- `n`: A numeric vector of length one identifying the number of observations in `x`. It is important the

user specify the number of observations so the `mean.release()` can touch the data as little as possible to ensure privacy preservation.

- `rng`: A numeric vector of length two identifying the range of `x`. If this vector is greater than length two, the maximum and minimum values of this vector will be used.

Examples

Basic Example

Attaching the sample dataset:

```
data(PUMS5extract10000, package = "PSIlence")
data <- PUMS5extract10000
```

Calculating the differentially private mean of a numeric vector using `mean.release`:

```
library(PSIlence)
private.numeric <- mean.release(x = data$income, var.type = "numeric", n = 10000,
                                epsilon = 0.01, rng = c(-10000, 713000))
private.numeric$release
```

```
## [1] 35701.36
```

To calculate the mean of a logical vector instead, input a logical vector into `x` and update `var.type` and `rng`:

```
library(PSIlence)
private.logical <- mean.release(x = data$married, var.type = "logical", n = 10000,
                                epsilon = 0.01, rng = c(0, 1))
private.logical$release
```

```
## [1] 0.5627687
```

Output Values

The output of the `mean.release` function stores fields containing the name of the statistic, variable type, number of observations, the sensitivity of the data, the selected epsilon value, the differentially private release of the mean, the accuracy guarantee of the mean release, and the bounds of the confidence interval for the mean release.