CMSC 23200 HW 7

Ruolin "Lynn" Zheng

February 27 2020

Part 1

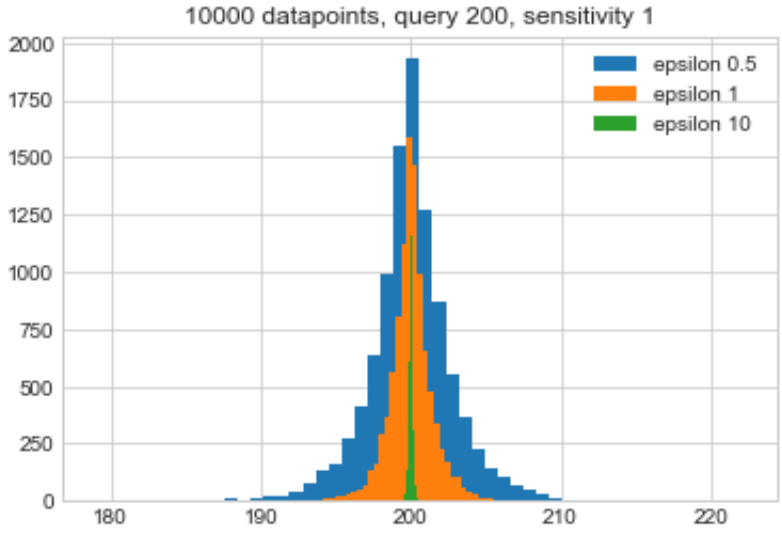
Problem 1.1

32560 individuals could be uniquely identified and one could not.

Problem 1.2

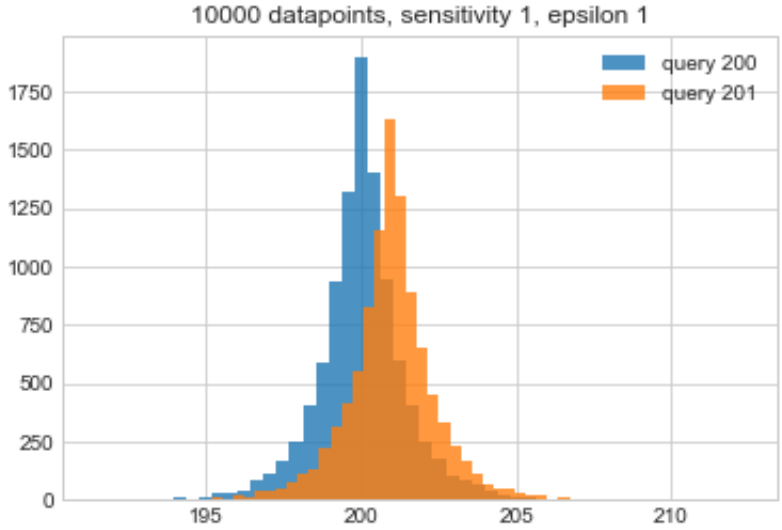
Suppose the number of rows is N, my algorithm will run in $O(N)$. My approach is to group by the given columns and count the number of unique tuples in each group. If the group with the smallest number of tuples has $>= k$ tuples, the table is k-anonymous. Otherwise it is not.

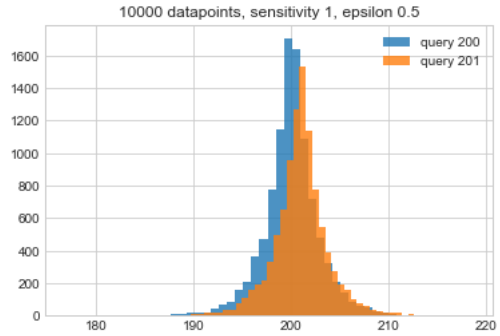
Problem 1.3

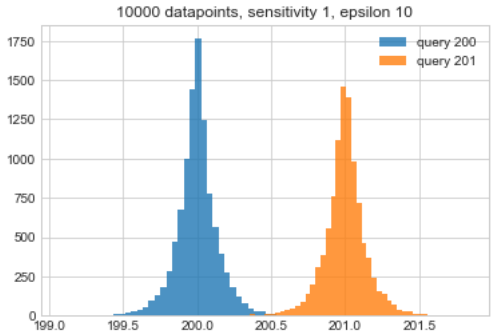


`num_bachelors` returned 5355. The sensitivity of `num_bachelors` is 1. $\Delta f = max_{D,D' neighboring} |f(D) - f(D^{'})| = 1$ when a person with a Bachelor's degree is in D but not in $D^{'}$.

Problem 1.4

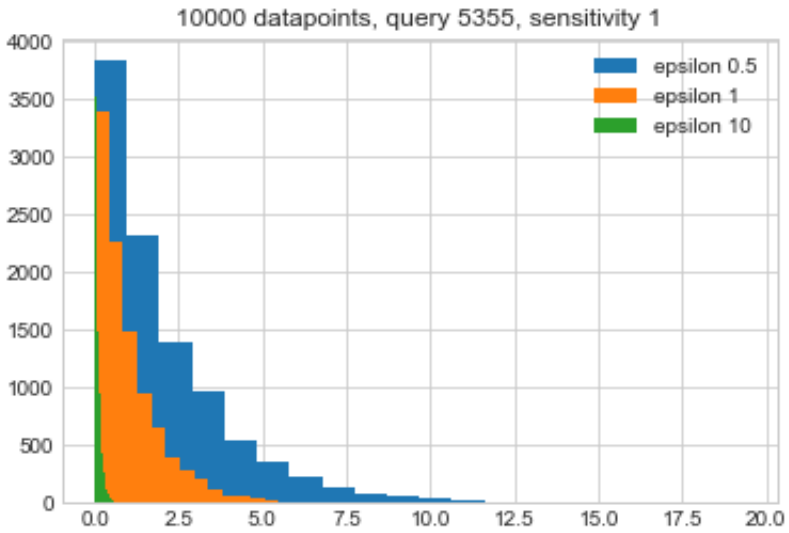






The greater amount of overlap between the histograms, the better privacy an individual in the dataset has. A lower epsilon value means more privacy, and by contrasting the plots of epsilons set to 0.5, 1, and 10, we observe that as epsilon decreases, there is more overlap.

Problem 1.5



As epsilon increases, the error decreases. The security/privacy of an individual in the dataset also decreases. We would want an epsilon value that is low enough to guarantee some privacy but high enough to preserve some accuracy of the query. In this problem setting, 1 might be the best candidate among 0.5, 1, and 10. We might devise some statistical tests and require that the error/noise in the query results be bounded according to some measures of the original data values. We might also bound the variance of the errors.

Part 2

Note: Please see `part2-plot.ipynb` for the plots used below.

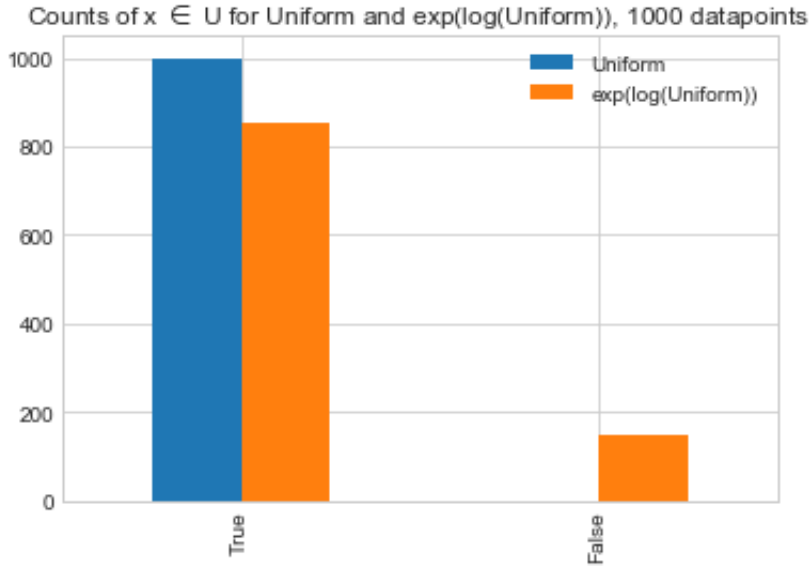
$Pr[m(D) \in S] \leq e^{\epsilon} Pr[m(D^{'}) \in S]$ no longer holds true since it's now possible that $Pr[m(D^{'}) \in S] = 0$. The implemented Laplace distribution would be missing some values because U doesn't output all possible doubles.

Instead of testing "any x $\in U$" as the writeup specifies, I found that I could more effectively drive down false positives from `1.+my_laplace(1)` by testing the membership of both $x = e^{y/s}$ and $x_2 = e^{\log x}$ (if $s * \log x_2 = -abs(y)$). I tested the membership of an element x in U by checking whether x is an integer multiple of 2^{-53}, as the paper states. I compute a double `x / (2 ** -53)` and an integer `x // (2 ** -53)` and return $x \in U$ if these two values are equal. My algorithm is very simple: return True if `x in U and s * log(x2) == -abs(y) and x2 in U` and False otherwise.

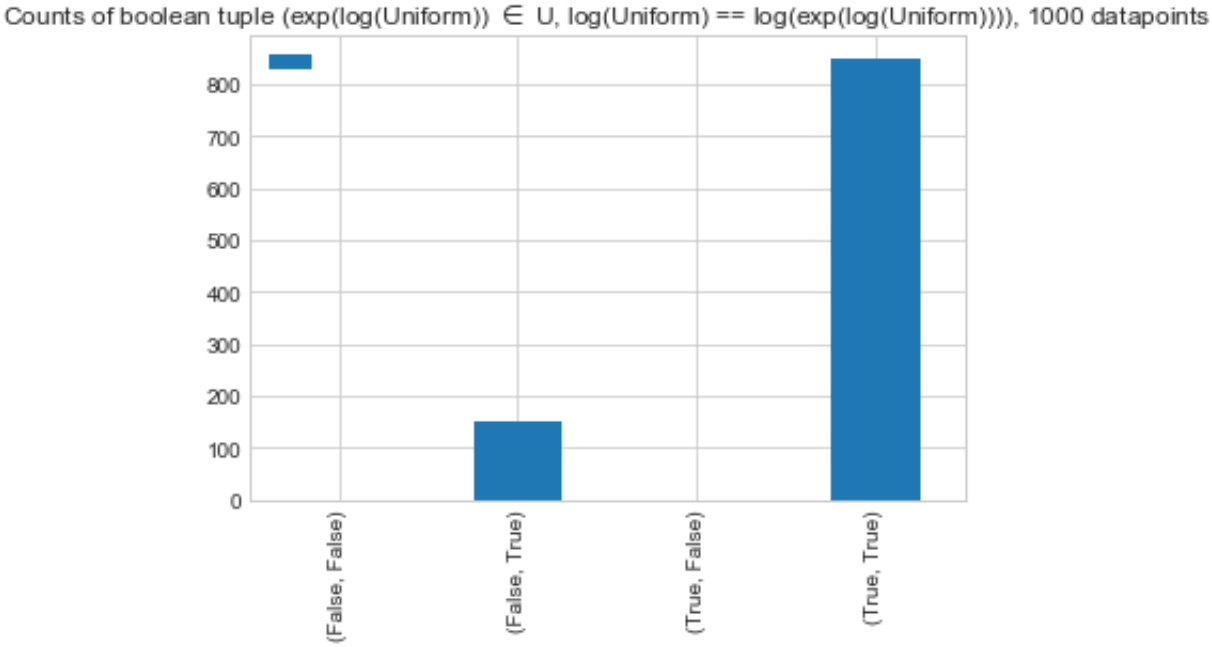
In one run, there were 850 true positives from `my_laplace(1)` and 337 false positives from `1.+my_laplace(1)`.

My attempt to offer a visual explanation of my approach is as follows:

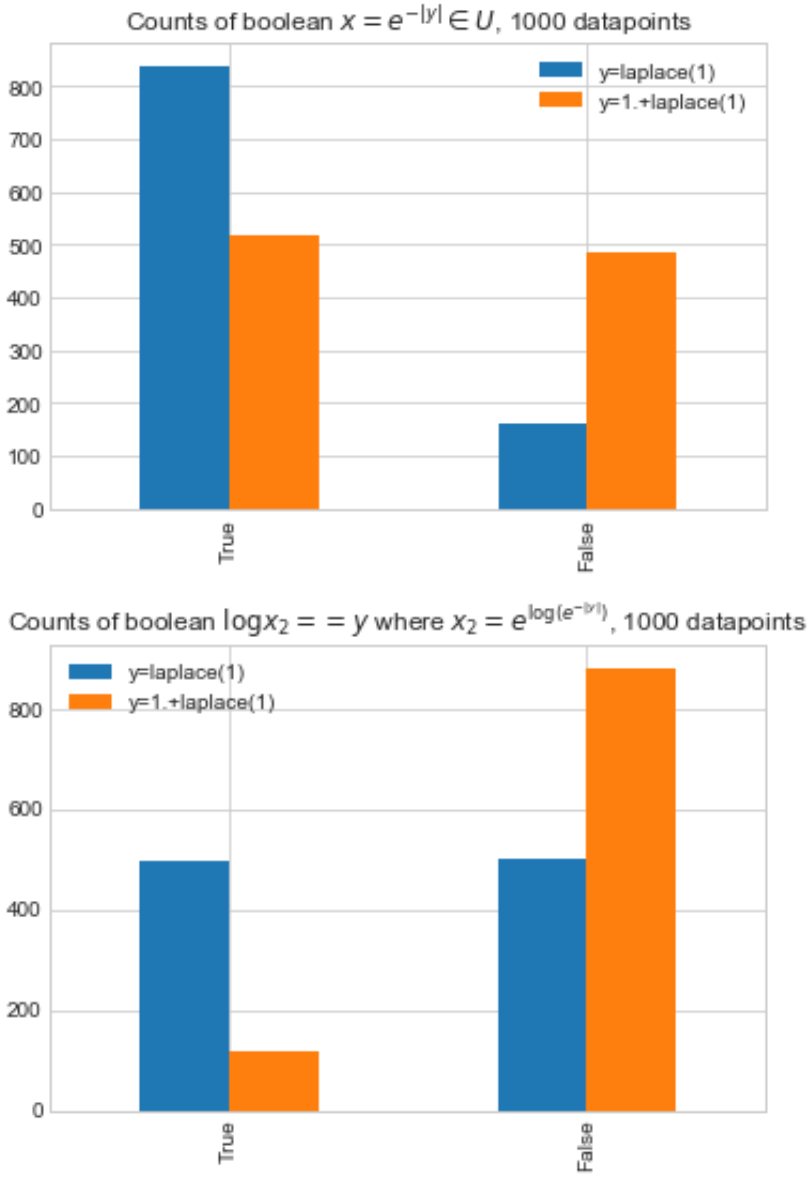
The plot below reveals that an x from `numpy.random.uniform` is guaranteed to be an integer multiple of 2^{-53} and belongs in U, whereas only about 80% of $x_2 = e^{\log(x)}$ satisfies $x_2 = x$ and belongs in U. This is because numpy's `exp` and `log` functions aren't exactly the inverse of one another.



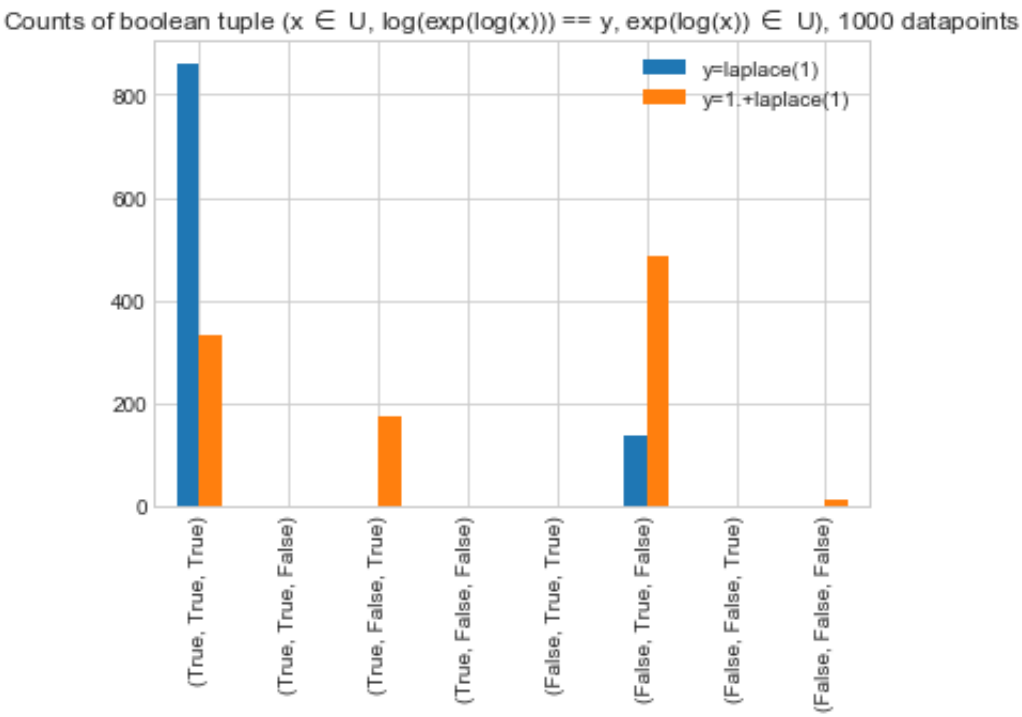
The plot below reveals that although only 80% of $x_2 = e^{\log(x)}$ belongs in U, $\log x = \log x_2$ always hold true for a uniform x.



Still using the notations $x = e^{-|y|}, x_2 = e^{\log(x)}$, the first plot below reveals that $P(x \in U) \approx 0.85$ when $y \sim Laplace(1)$ and $P(x \in U) \approx 0.5$ when $y \sim Laplace(1) + 1$. The second plot reveals that $P(\log x_2 = y) \approx 0.5$ when $y \sim Laplace(1)$ and $P(\log x_2 = y) \approx 0.15$ when $y \sim Laplace(1) + 1$. These great differences in probabilities led me to evaluate the boolean tuple $(x \in U, \log x_2, x_2 \in U)$ as in the final plot.



Observe that $P(x \in U, \log x_2, x_2 \in U) \approx 0.85$ when $y \sim Laplace(1)$ whereas $P(x \in U, \log x_2, x_2 \in U) \approx 0.3$ when $y \sim Laplace(1)+1$. The difference in the number of Trues could be around $(0.85-0.3)*1000 = 550$.


Counts of boolean tuple (x ∈ U, log(exp(log(x))) == y, exp(log(x)) ∈ U), 1000 datapoints
y=laplace(1)
y=1.+laplace(1)
800
600
400
200
0
(True, True, True)
(True, True, False)
(True, False, True)
(True, False, False)
(False, True, True)
(False, True, False)
(False, False, True)
(False, False, False)