Final Report: Visualization of Uncertainty in Differential Privacy

Jessica Bu

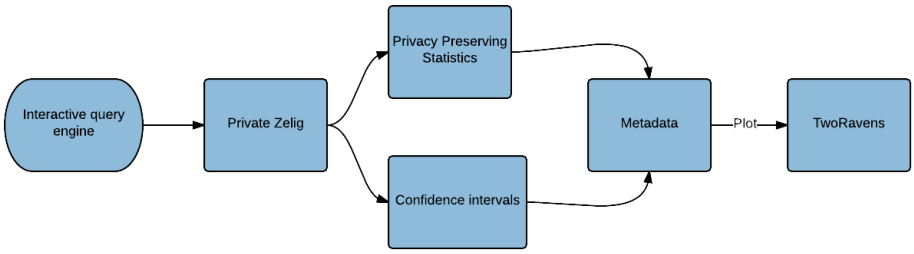
Harvard REU Summer 2015

Introduction

The idea behind differential privacy, a method of anonymization, is to ensure that results derived from a database look the same whether or not it contains a given individual's data. Differential privacy works by adding precisely constructed random noise to the data such that statistical results do not reveal any individual-level information, while remaining useful to researchers. What this means, however, is that there is an inherent degree of uncertainty associated with the statistics released by differentially private mechanisms.

My goal for the summer was to visualize the uncertainty introduced by differential privacy, specifically in TwoRavens, a graphical user interface targeted at users of all levels of statistical expertise . It integrates with the statistical software package Zelig, and the data repository Dataverse, allowing users to explore and run statistical models on their data. When handling a dataset containing sensitive information, TwoRavens should be able to communicate to users that the answers displayed are not necessarily the true values but have noise added to them. Conveying uncertainty is important to avoiding misleading researchers and drawing incorrect conclusions due to a lack of understanding of the reliability of the information. We would like to therefore include uncertainty whenever displaying differentially private statistics in TwoRavens.

Visualization Pipeline



Interactive query engine:

A researcher uploads or selects a dataset in Dataverse, chooses global parameters, and specifies queries to make about the data using the interactive query engine.

Private Zelig:

Back-end R-libraries in a differentially private version of Zelig are accessed, which contain functions for calculating privacy preserving statistics and their confidence intervals.

Privacy preserving statistics:

There are currently three types of statistics that are released by differentially private mechanisms inside the R-libraries: means, histograms, and quantiles. Since the structure for each of these releases was being written differently, I rewrote the functions to be called in the same way:

***statistic*.release(eps, data, ...)**

Input:

eps – epsilon privacy parameter
data – vector of data
remainder of arguments dependent on type of statistic

Output:

A list of two variables: release and params. [1.5em] **release** – privacy preserving statistic
params – list of the parameters that were passed into the release function (eps, del, etc.), in addition to n (the number of elements in the data), but excluding data.

Confidence intervals:

The confidence intervals calculated describe the uncertainty associated with a privacy preserving version of the true sample value, and are defined as the range that captures some defined fraction, commonly 95 percent, of the probability distribution of the underlying value. That is, if we drew 100 differentially private releases for a sample answer and constructed 95% confidence intervals for each release, we would expect 95% of them to contain the true sample answer.

I wrote a function to construct confidence intervals for each statistic that is ready to be released, that is, for means and histograms. These functions construct confidence intervals from the differentially private release and the parameters passed into the release function, which include the privacy parameters (eps, del), in addition to n (the number of elements in the data), but do not

touch the data itself. Both the release and the parameters are returned as elements in a list by the release function.

***statistic*.getCI(release, params, alpha=0.05)**

Input:

release – the privacy preserving statistic released by the release function
params – the parameters outputted by the release function
remainder of arguments dependent on type of statistic
alpha – maximum failure probability (default: alpha=0.05)

Output:
A list of pairs of upper and lower confidence limits.

Metadata:

After the values for the privacy preserving statistics and confidence intervals are generated, they are written into a JSON metadata file, which contains two objects at the top level: "dataset" and "variables." Dataset contains a boolean property "private," which states whether or not the dataset handles private information. "Variables" contains all of the preprocessed information data for each variable in the dataset.

TwoRavens:

TwoRavens (app_ddi.js) accesses the metadata file. If "private" is true, it checks for confidence intervals and if they exist, reads in the values and displays them according to the type of the statistic, as detailed below.

Means:

Confidence intervals for means are appended as numerical values. The mean of a variable is displayed as part of the summary statistics in the Summary tab when a user hovers over its pebble (Fig. 1), and in a popover upon mouseover of its name in the Variables tab (Fig. 2).

Figure 1: Hovering over the pebble for the age variable (in the center space) displays the differentially private mean and its 95% confidence interval in the Summary tab (left panel)

Figure 1: TwoRavens using a PUMS dataset, n= 2000.
Hovering over the pebble for the age variable (in the center space) displays the differentially private mean and its 95% confidence interval in the Summary tab (left panel)

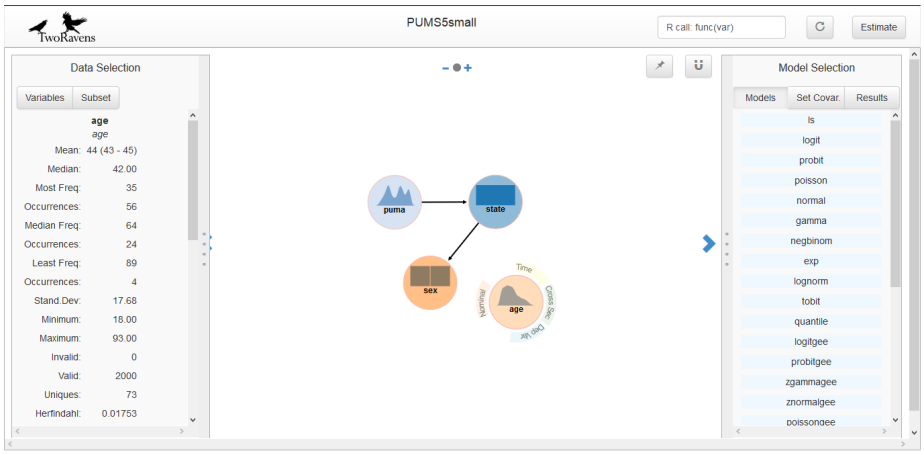
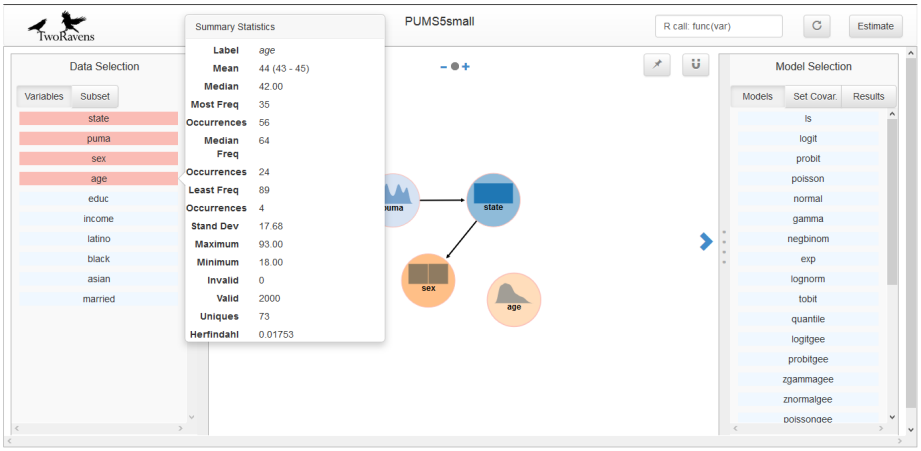


Figure 2: TwoRavens using a PUMS dataset, n= 2000.
Hovering over a variable name in the Variables tab displays a popover containing the summary statistics. Again, the mean is displayed with a 95% confidence interval.



Histograms

The file containing Histogram.getCI (HistogramCI.r) also contains two other getter functions for stability based histograms, Histogram.getThreshold and Histogram.stabilityBin.

Histogram.getThreshold returns the threshold value for histograms released by the stability based algorithm – any bins whose counts fall below the threshold are excluded from the perturbed histogram. When we plot histograms, the threshold value should be displayed to make sure the users know that the non-zero bins near the threshold could have likely been set to zero instead.

Histogram.stabilityBin gives an estimate of the sum of the counts that were set to zero in a stability based histogram. An extra bin with the height of the value returned is appended to the end of the histogram to give users an idea of the bins that were excluded. The function returns the difference in the number of elements in the dataset and the sum of all the counts that are displayed.

Histogram.getCI(params)

Input:
params – the parameters outputted by the release function

Output:
A single value, the threshold. noisy counts below the threshold are set to zero.

Histogram.stabilityBin(release)

Input:
release – the stability based histogram released by the release function

Output:
A single value, the difference between the size of the sample and the sum of all of the counts of the stability based histogram.

Displaying Uncertainty in Histograms

Error bars are used to represent the confidence intervals for histograms, which are displayed for variables whose "plottype" value is "bar" in the Summary tab upon mouseover of their pebbles (Fig. 3).

Figure 3: TwoRavens using a PUMS dataset, n= 2000.
Histogram with error bars representing 95% confidence intervals for the sex variable.

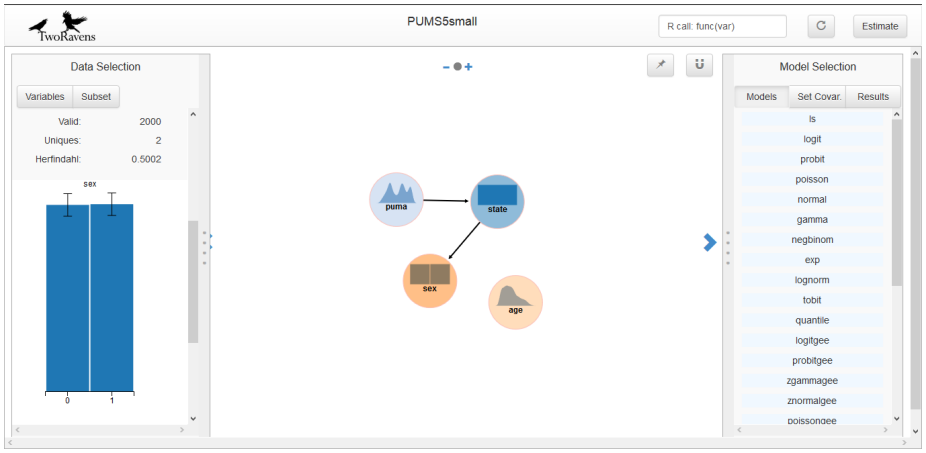
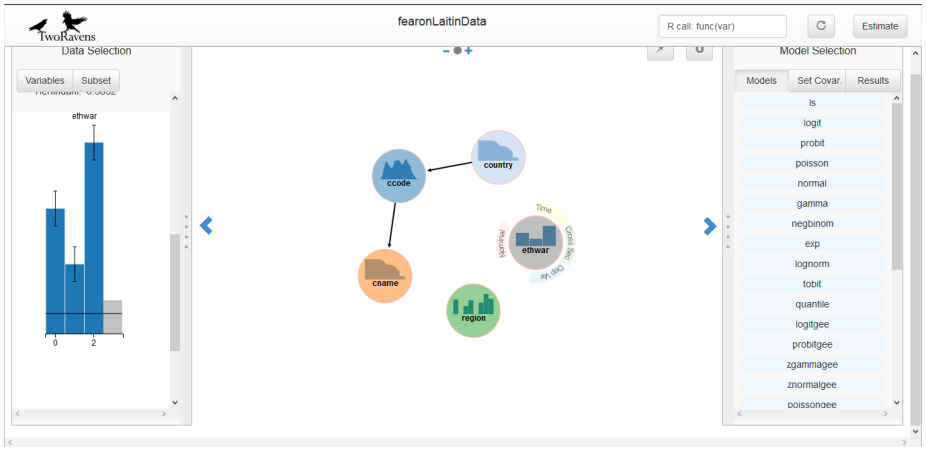


Figure 4: TwoRavens using a fearonLaitin dataset, with values for the threshold and stability bins hardcoded into the metadata file for the variable ethwar (values generated by the getThreshold and stabilityBin functions are not currently being written into the metadata file). It is clear that no counts were actually set to zero, but this graph shows what it would look like if any bins had needed to be excluded.



Dense Histograms

Figure 5: Histogram for the variable country. Because there are so many bins, the error bars completely black out the top portion of the histogram.

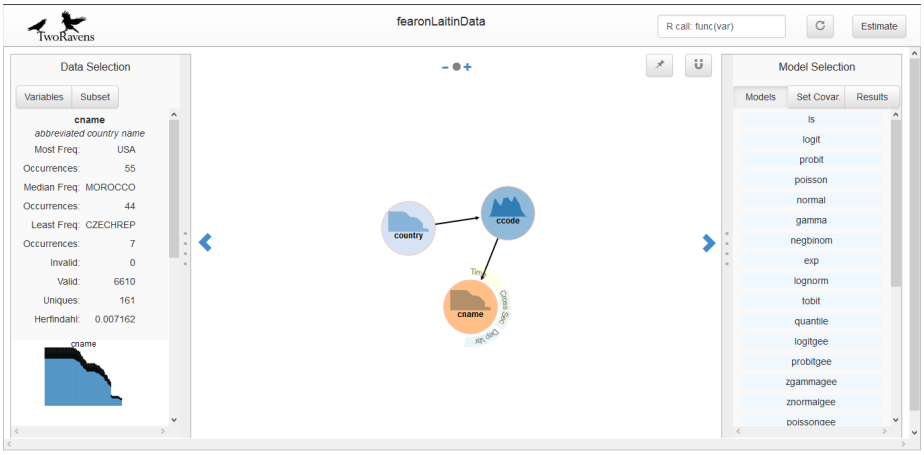
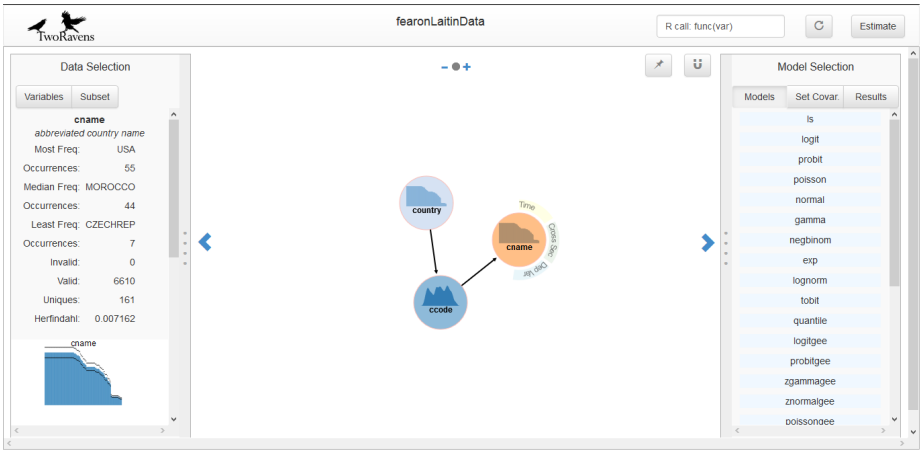


Figure 6: Variation: For any histogram with number of bins exceeding 20 (number decided somewhat arbitrarily), the vertical bars are removed from the error bars, leaving only the top and bottom ticks.



While this variation of the error bars works here because the ticks seem to form an almost continuous line, it does not work as well for graphs with steeper bin differences:

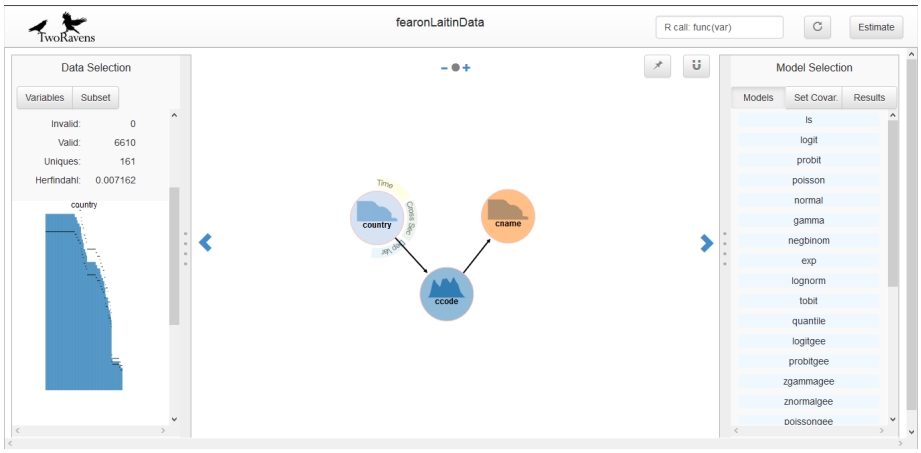
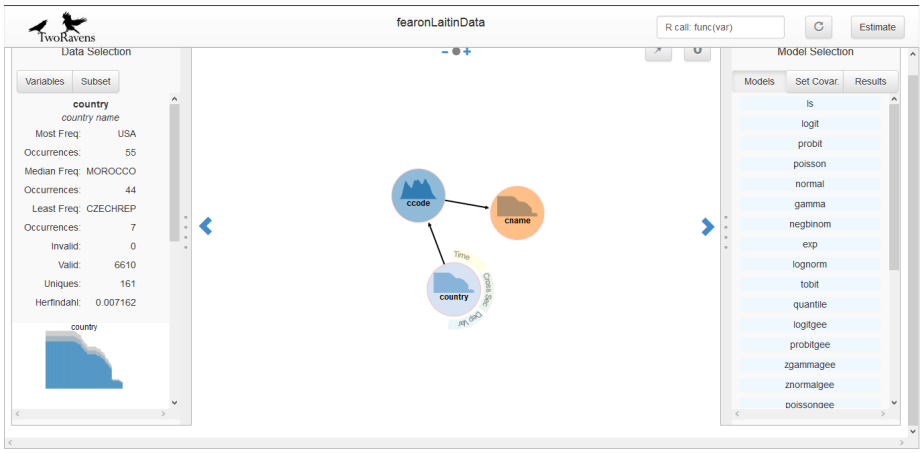
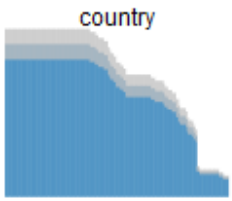


Figure 7: Final variation

(a) Each error bar is replaced completely by a rectangle shaded in grey, spanning the length of the 95% confidence interval.



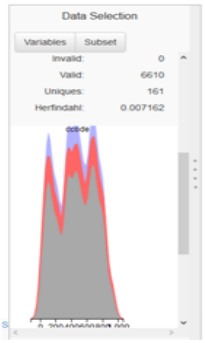
(b) A closer look at the dense histogram



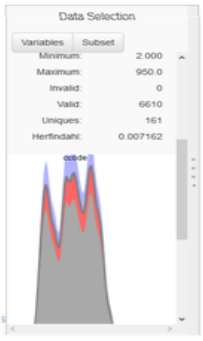
Density Graphs

As of right now, the differentially private R libraries do not contain a function for releasing a density graph, but Figure 9 shows what a density graph ("plottype": "continuous") would look like with uncertainty.

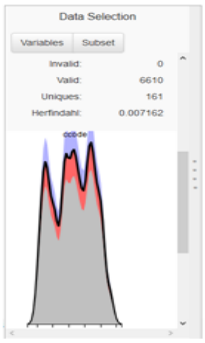
Figure 8: (A selection of) the graphs created during the design process. In each of these graphs, the area between the uppermost and lowest curves represents the 95% confidence intervals at each point on the density graph.



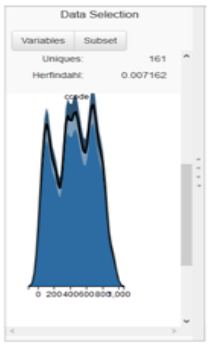
(a) There seems to be three different distributions, and it is unclear which is actually the density graph.



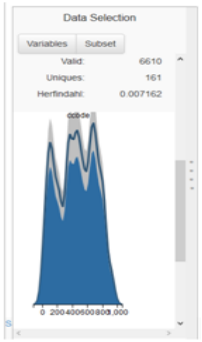
(b) An attempt at emphasizing the density graph; however, there is a disconnect between the line and the visible portion of the density graph.



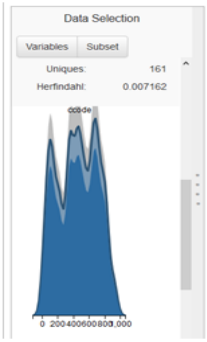
(c)



(d) Medium shade of blue to represent the density graph, lighter shade of blue to represent the lower half of the confidence intervals, and darker shade to represent the upper half of the confidence intervals.



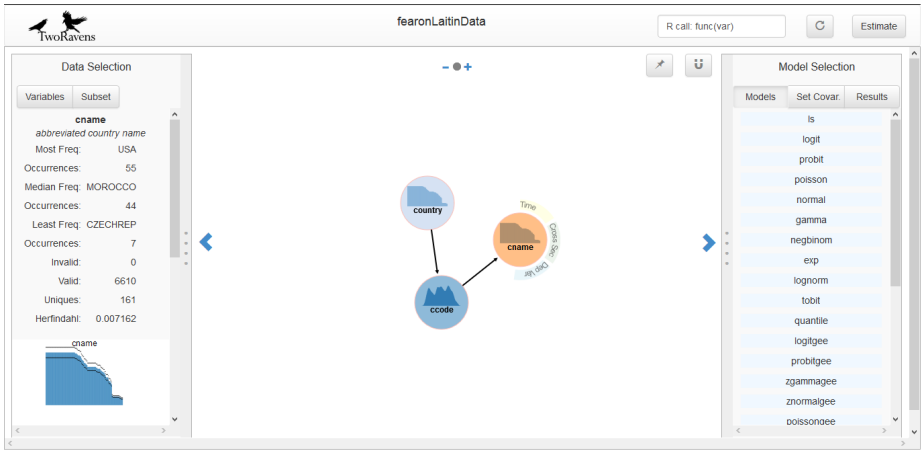
(e) Uniform shade of grey to represent confidence intervals. However, there seems to be a disconnect between the edge of the density graph and the visible, blue portion.



(f) Attempt to fix the disconnect: the bottom half of the confidence intervals is a lighter shade of blue to make it seem like the blue of the density graph is overlaid with the grey of the confidence intervals.

The current graph being used is shown in Figure 9. The colour scheme is blue to fit in with the graphs of all the other types of statistics, which are also blue. We settled on this graph because the density graph can be seen in its entirety, so there is no disconnect between the edge and the portion underneath the lower confidence limit. The upper and lower halves are both grey to convey that they represent the same thing, ie. the confidence intervals; since the grey area is slightly transparent to allow users to see the density graph underneath, the bottom portion of the confidence intervals appears to be a light shade of blue.

Figure 9: Density graph with error for the age variable.
The values of the confidence intervals are $\pm$ 20% of the original values, hard-coded into the metadata file and read into plots.js.

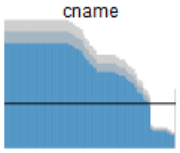


Conclusion

We were able to accomplish the goal that we set for ourselves: to communicate the noise for differentially private releases.

Future Work

- getCI functions should be written for other types of releases as they become available
- There is an issue with error visualization in stability based histograms: in histograms with a large number of bins, the additional bin representing the unaccounted for bins is very thin:



- We visualized the error in the full versions of the graphs, displayed in the Summary tab, but the graphs also appear in the pebbles. How could you create a small scale version that communicates uncertainty without being too visually cluttered?
- It could be useful to allow researchers who are uploading their data to see what the uncertainty looks like.
- Although clarity and ease of understanding were kept in mind while designing the uncertainty visualizations, the density graphs could still be confusing. It might be more intuitive to use line charts with confidence interval areas instead of area charts.
- Usability testing: explanations still need to be added describing what the error bars, etc. represent. Once that is done, it would be useful to ask users how clear they find the uncertainty visualizations and how easy to understand/interpret. Other, more specific questions should be asked to test their understanding. For example, why is the last bin (in a stability based algorithm) so full?