Lancelot Wathieu (REU 2017 intern)

[lbw29@georgetown.edu](mailto:lbw29@georgetown.edu) or [lbw2148@columbia.edu](mailto:lbw2148@columbia.edu)

**Final Report Privacy Tools**

**Introduction**

Social scientists and companies such as Google, Apple, and Uber often want to analyze sensitive data that may contain private information about individuals. The Privacy Tools Project aims to build statistical tools that give close-to accurate answers to queries on a dataset while preserving the privacy of each individual in that dataset. These tools satisfy “differential privacy” – they come with a guarantee that the answer to the query is almost the same whether or not a particular person is in the dataset. PSI (Private data Sharing Interface) is a system to allow researchers to share and explore sensitive datasets with differential privacy (<https://arxiv.org/abs/1609.04340)>. With this tool, researchers receive plenty of the utility of raw data, and no information about individuals is leaked. This summer I have worked to improve causal inference as one of the statistics available to users of the tool.

I have worked on implementing a differentially private coarsened exact matching. Coarsened exact matching (CEM) pre-processes observational data in order to better perform causal statistics. CEM matches treated and non-treated individuals using as many confounding variables as possible, in order to isolate the effect of a treatment on some outcome. Normal causal techniques, like difference of means, can then be used on the processed data to perform causal inference.

This work aims to implement CEM in PSI. To make CEM interactive and differentially private, the Sparse Vector Technique is used. This technique is used to give users of PSI more power over how ‘good’ their matching of the data is, while keeping the data differentially private. We hope to integrate both interactive and automatic versions of coarsened exact matching into PSI.

**Coarsened Exact Matching**

**(**[**King’s CEM R library**](https://gking.harvard.edu/cem) **and** [**King’s CEM paper**](https://gking.harvard.edu/files/political_analysis-2011-iacus-pan_mpr013.pdf)**)**

**The Need**

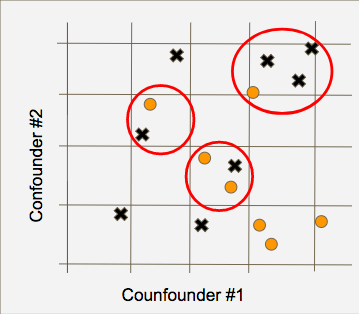
Causal Inference studies the effect of a treatment on some outcome variable. Often, social scientists who want to study causal inference must conduct randomization studies, which can be expensive or unethical or both. However, due to its abundance it is very useful to be able to use observational data to study causality. Matching is a technique of pre-processing observational data in order to derive more accurate results when performing statistical analyses like difference of means. The difference of means is used to estimate the effect of the treatment on the outcome.

The goal of matching is, for every ‘treated’ individual in a dataset, to find one or more non-treated units with similar observable characteristics so that the effect of the treatment variable can be isolated from confounding factors. In coarsened exact matching, each continuous ‘confounding’ variable is coarsened into bins and any units that are unmatched are thrown away. By ‘matched’ in this case, we mean that for every treated unit there is one untreated unit within that stratum (‘one-to-one’ matching).

**Exploration of Coarsenings**

There is a tradeoff when we do coarsened exact matching. If the bin widths are too small, then two individuals within one stratum will be very similar, resulting in a good matching. However, more treated units will be left alone without untreated units within a stratum, so we will end up losing a lot of data points. If the bin widths are very large, we will retain plenty of data but two units within a stratum might be drastically different (resulting in data that is hardly matched at all).

A coarsening is one particular set of confounding variables and their attributes. A coarsening includes a list of scale variables (along with their binwidths), a list of group variables (along with their groupings), and a list of exact variables (these terms are defined below). As intuition might lead us to believe, as we add variables to this coarsening, the strata become higher-dimensional and the number of treated variables remaining after matching decreases. The number of treated variables remaining also decreases as the binwidths decrease for scale variables and as groups become less inclusive.



**Figure 1**

Figure 1 shows points of a dataset plotted using two confounders. The binnings of the variables are a coarsening, where circles are treated and crosses are untreated. Any points outside the red circles are unmatched within each stratum and so would be removed.

Iacus, King, and Porro have previously built the R Package “CEM: Software for Coarsened Exact Matching”. Our differentially private CEM is largely formatted based on this package. In our code, confounding variables are classified as scale, group, or exact variables.

Scale variables are the variables that we usually think of when coarsening. These are variables for which only a bin width is specified. As such, continuous variables such as age and income are usually classified as scale variables. For example, we might set 30,000 as a good binwidth for annual income. Then, anyone with annual income $0-$30,000 are considered the ‘same’, income $30,000-$60,000 are considered the same, and so on.

Group variables are variables for which discrete values can be grouped together into categories. Categorical variables such as race or level of education are frequently classified as group variables. If education is on a 1 to 10 scale, we can specify different groupings of the values within which we consider units to be the ‘same’. For example, we can group levels of 1,2,3 together (consider all less-educated people the same), levels of 4,5,6,7 together (consider all educated people the same), and levels of 8,9,10 together (consider all highly-educated people the same).

Exact variables are variables for which we match exactly. We consider two people the ‘same’ only if they have the same value. This usually is best for discrete variables such as gender, but can be used for categorical or even continuous variables if sample size if extremely large.

In non-private CEM, the input is a dataset and a coarsening (with scales, groups, and/or exact), and the output is a matched dataset, as well as the number of treated units remaining. For a differentially private version, we could do the same process but add noise to the number of treated units. This is what we might consider a DP one-shot CEM. However, much of the utility of CEM comes from a user’s judgement of what a ‘good match’ means. This means that getting a feedback about how changes to the coarsening affects the remaining number of treated units is important. With the sparse vector technique, we can add some of this flexibility while keeping the process differentially private.

**DP CEM: 3 different strategies**

1. **One-shot CEM**

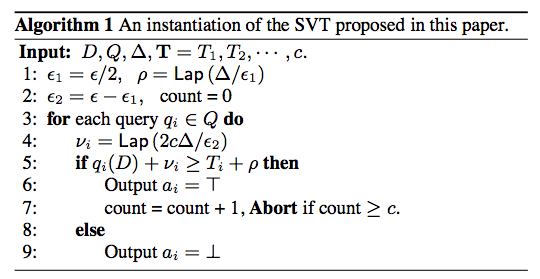
In one-shot CEM, the user inputs a dataset, a coarsening and a matching type (in this case one-to-one). The algorithm outputs a noisy number of treated units after matching. Within each stratum of the coarsening, the algorithm checks if there are any treated-control pairs. Any points that do not have a pair within the strata are removed. Then, the remaining treated units are counted and that number is returned with some Laplace noise added.

1. **Interactive CEM**

For an interactive version of coarsened exact matching, the Sparse Vector Technique is used.

**The Sparse Vector Technique (**[**arXiv:1603.01699**](https://arxiv.org/abs/1603.01699)**[cs.CR]**)

The Sparse Vector Technique (SVT) is a technique for satisfying differential privacy in an interactive setting. SVT allows us to answer a sequence of Yes/No queries where we end up paying a privacy cost only for those queries where the answer is yes. Given a threshold T and a sequence of queries, SVT compares T and the query answers (both with noise added) and outputs a vector of whether the queries were above or below the threshold. SVT only uses privacy budget for query answers on one side of the threshold. If the goal is to find a query that returns an answer above the threshold, we can ask as many queries as we want that will return an answer below the threshold (a ‘negative’ answer). The privacy budget is only spent on query answers that result in an answer above the threshold (a ‘positive’ answer). Extraordinarily, we can base our subsequent queries on SVT feedback of previous queries. Both the query and the threshold can be changed during each iteration.



**Figure 2**

Figure 2 shows an instantiation of the SVT algorithm proposed in ([arXiv:1603.01699](https://arxiv.org/abs/1603.01699) [cs.CR]). Here, D is private database, Q is a stream of queries, T is a sequence of thresholds, c is the maximum number of queries to be answered with a positive answer.

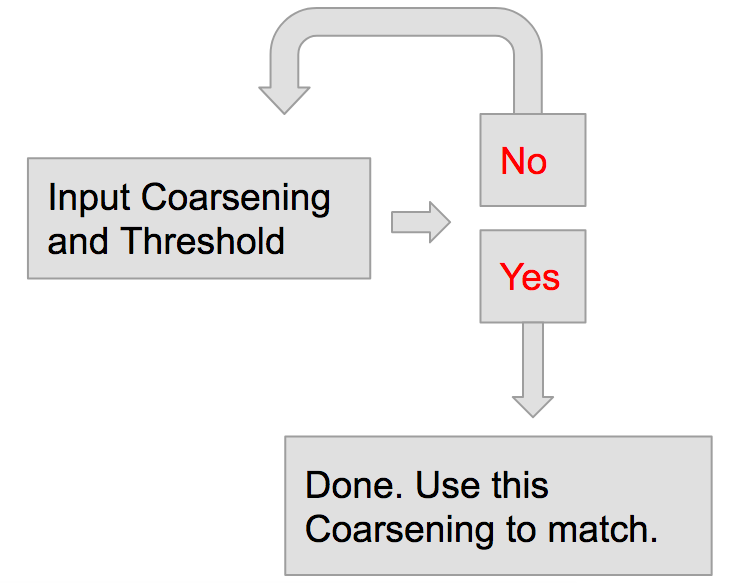
ai is a query answer (, where denotes a ‘positive’ and denotes a ‘negative’ answer

**Using SVT in an interactive CEM**

To use SVT in CEM, we have to define what the queries are and what the threshold is. The queries ask how many treated units (with noise, denoted nT) a specific coarsening gives after matching. As the threshold for SVT, we are using the desired number of treated units nTD. The inputs are a coarsening and nTD, and the output is a Boolean; either nT is below (‘no’) or above (‘yes’) the nTD.

Given an unmatched dataset’s noisy number of treated units, we assume that the user has a good idea of how many units he wants to keep. On one hand, they want to keep nT small so that the matches are good. On the other hand, they do not want to lose too many datapoints. The user first inputs his nTD and specifies an initial coarsening of the variables. Ideally, this initial coarsening is fine enough so that nT < nTD (output: ‘no’). In this case, the user can adjust the coarsening (by relaxing it in some way) and/or nTD by decreasing it. The user can try as many combinations as he wants, and only pays in privacy for each ‘yes’ response.

With this interactive algorithm in combination with an intuitive user interface, we hope to create an easy way for a user to have more control over their matching.

****

**Figure 3, layout for interactive CEM**

1. **Automatic-Relaxing CEM**

Auto-CEM is useful if a user does not want to go through the trial-and-error process of the interactive CEM and trusts their judgement of nTD. Auto-CEM uses an initial user-provided coarsening and nTD. The algorithm then stretches the bins of scaleList systematically until nT > nTD. This algorithm also uses SVT, since it only sends queries that have ‘no’ answers until finally it stops when the last query has a ‘yes answer. The groups and exact variables of the initial coarsening are not changed, neither is the threshold (nTD).

**DP CEM: Tests and Evaluations**

**Datasets**

For possible testing of CEM and how it works with causal statistics, I found a great causal dataset (from the Privacy Tools replication study corpus). The study looks at whether automated calls influence voter turnout or vote choice. The sample size is ~2.5 million, with ~15,000 treated units. The data has 10 confounding variables, one of which is continuous (age).

* *Do Robotic Calls from Credible Sources Influence Voter Turnout or Vote Choice? Evidence from a Randomized Field Experiment* (Gerber et. al, JPM, 2015)
* Article - <http://isps.yale.edu/research/publications/isps12-022#.V356jY6pJrY>
* Dataset - <http://isps.yale.edu/research/data/d044#.V3FZko6pJrY> , but I used the processed data from <http://isps.yale.edu/research/data/d026>

This dataset could be an ideal test dataset for CEM in conjunction with causal statistics, because of the large number of treated units and the large number of confounders. Because it only has 1 continuous confounder, it is not a great fit for auto-coarsening.

**Evaluation of Auto-CEM**

To evaluate the Auto-CEM algorithm, the DP auto-CEM was run many times with different threshold values. Then, histograms of the DP recommended % to stretch the scale variable binwidths are drawn against the single non-DP auto-CEM stretch recommendation. Histograms of the DP resulting number of treated units after matching are also plotted against the single non-DP auto-CEM number of treated units retained.

Parameters:

**#M** is the number of times to run the non-DP auto-coarsen. We use 300. increase to get more accurate histograms.

**#ep** is the overall epsilon value. Lower is more secure, but more noise is added. We try ep=1, ep=.5,ep=.2. This epsilon is split between 5 epsilons

**#keep** is the % treated units retained rate. We 50,55,60,65,70,75,80,85.

**#matchType** = "onetoone"

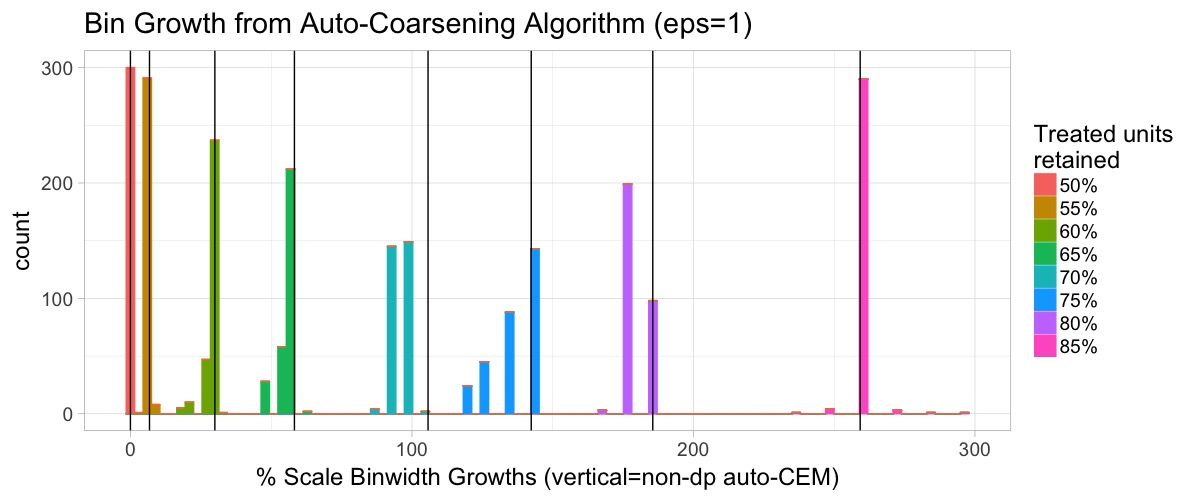
**Evaluation**

The ‘a’ graphs show how much ‘stretching’ of the bins the algorithm recommends.

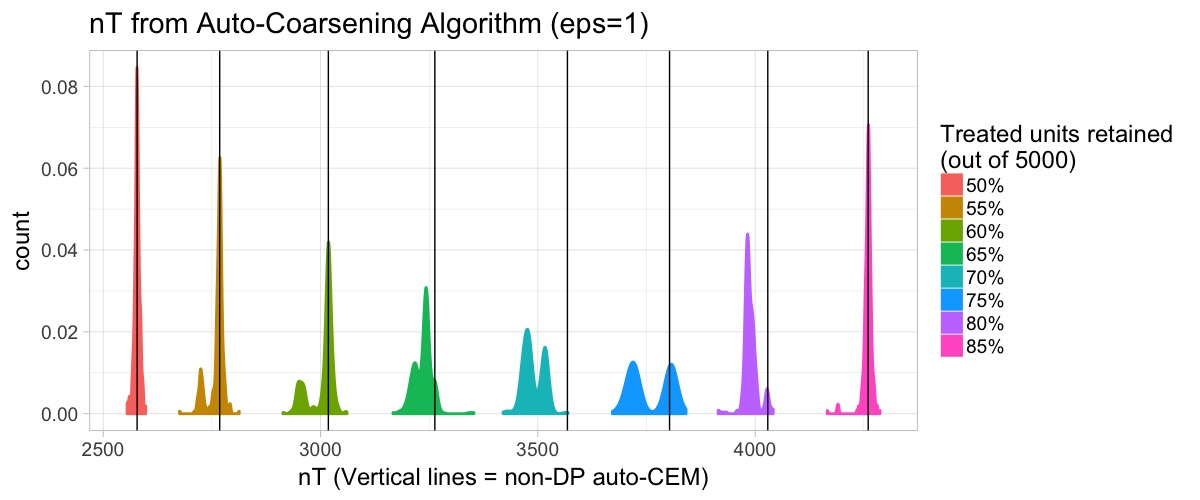
The ‘b’ graphs show how many treated units remain after the algorithm runs.

These are histograms of 300 different runs of the DP auto-coarsen. The non-dp answers are the vertical lines

**Epsilon = 1.0**

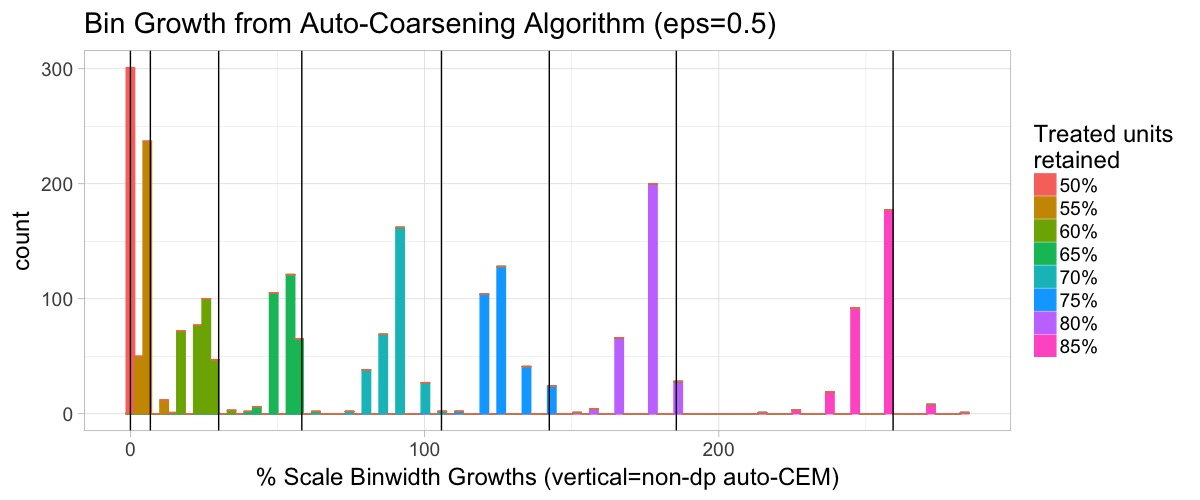


**Figure 4a.**

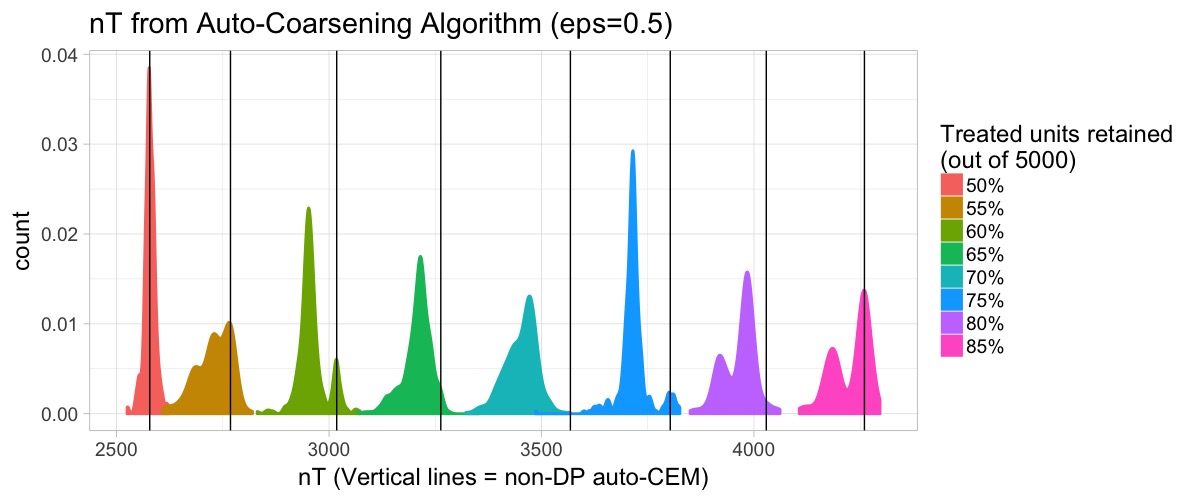


**Figure 4b.**

**Epsilon = 0.5**

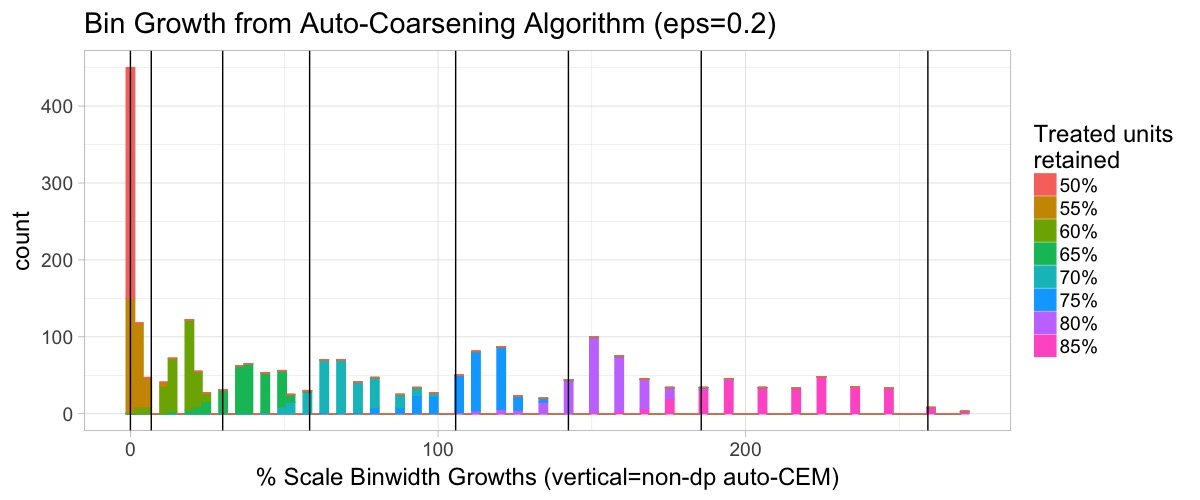


**Figure 5a.**

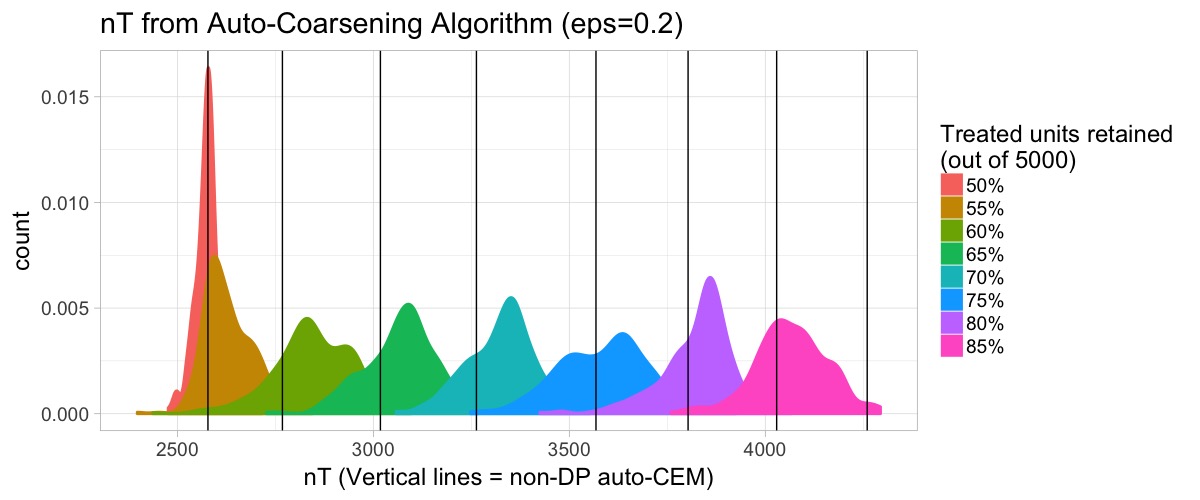


**Figure 5b.**

**Epsilon = 0.2**



**Figure 6a.**

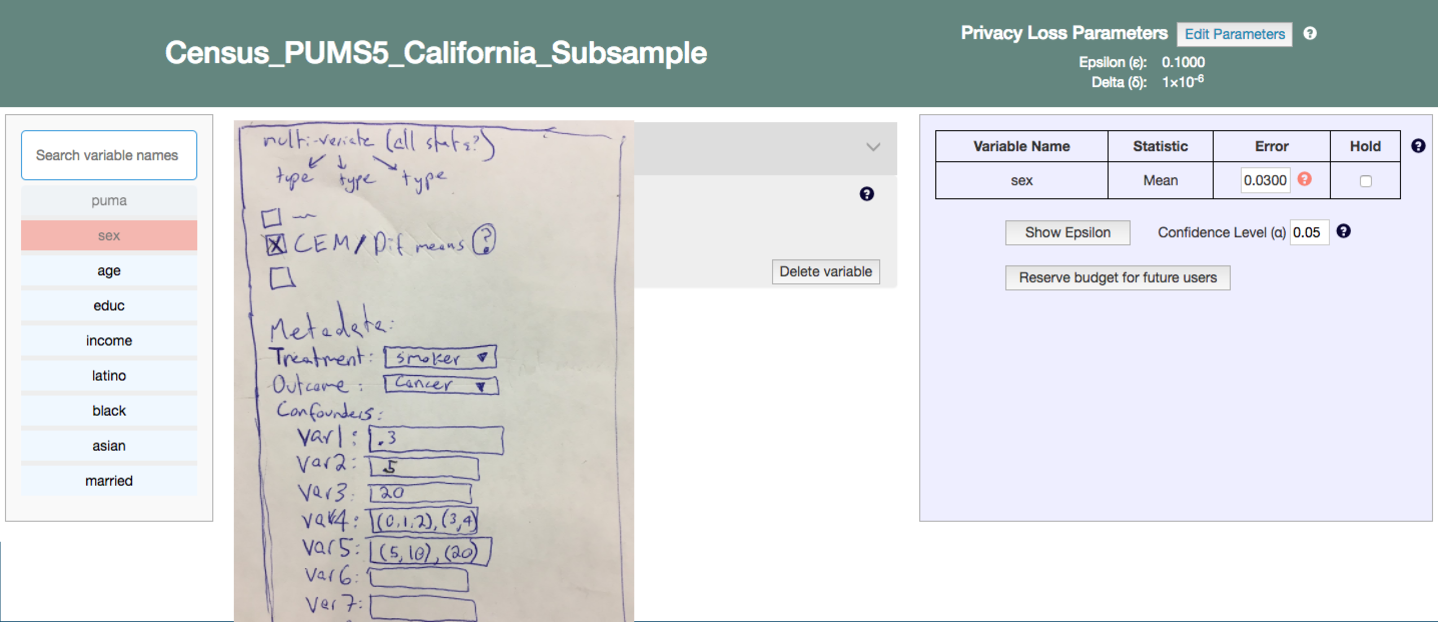


**Figure 6b**

Figures 4, 5, and 6 all show that the current iteration of our DP auto-CEM tend to coarsen the bins too little (the bins were finer than the non-DP version. As a result, the number of treated units was also too coarse. It’s important to note that these nT have noise added to the final answer as well. Also, as epsilon decreases the histograms are wider, as expected. However, they also become more under-estimated as epsilon increases. Note that in 6b, the distributions are shifted so far left that they overlap with the wrong vertical lines.

**Integration into PSI**

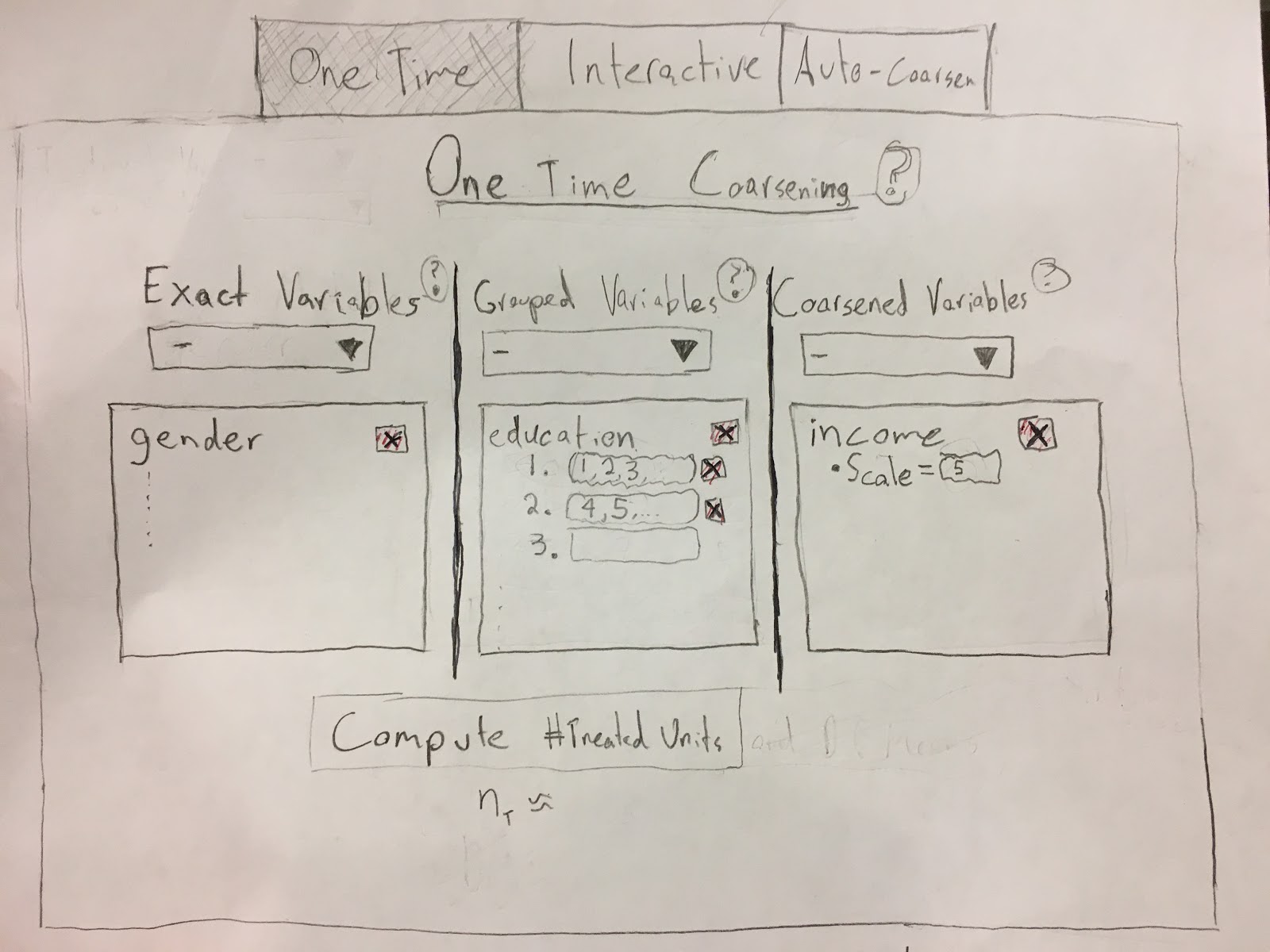
Originally, I had hoped to integrate an interactive CEM interface into PSI’s current budgeter prototype ([beta.dataverse.org/custom/DifferentialPrivacyPrototype/UI/code/interface.html](https://beta.dataverse.org/custom/DifferentialPrivacyPrototype/UI/code/interface.html)) as a multivariate statistic. However, problems arose when I realized that the current budgeter does not allow interactions with the data within a statistic. The architecture allows the tool to touch the dataset only once all statistics are chosen. To get around this problem and as a proof of concept, I thought it would be a good idea to have a 1-shot CEM and difference of means combo as a multivariate statistic in PSI. Figure 7 show a concept of how CEM/DoM could be easily set up as a multivariate statistic in PSI’s current budgeter.

****

**Figure 7**

In figure 7: var1, var2, and var3 are scale variables, var4 and var5 are group variables, and var6 and var7 are exact variables. The metadata input of each confounder could be parsed to create a coarsening that can be matched on and subsequent difference of means can be done.

The next implementation idea is a full GUI tailored for CEM. Figure 8 is a concept for this CEM GUI. [My final presentation](https://docs.google.com/presentation/d/17xD9A26IJNFVLOdQVPiNCIKd3uhC54Kh7Ps_V3_gf3g/edit#slide=id.g24cdf64a93_0_0) (slide 6) shows this model in more detail, with google slide transitions.



**Figure 8**

**Current Problems with CEM implementation**

* CEM 1-shot and automatic functionality is still being built into the PSI R package
* Figure 7 was never implemented because PSI does not currently support multivariate statistics (Should be available by Fall 2017). Also, is 1-shot CEM very helpful?
* Instead of tailoring a part of PSI for CEM directly, does it make sense to add some sort of separate pre-processing option to PSI? Matching can be used as a pre-processing step for many statistics, so it should be separate from difference of means. Other pre-processing steps could then be added to the PSI option, for example possible missing data imputation or other types of subsetting of a dataset.
* In what sort of setting would a figure 8 type model be implemented?
* What are good data-independent accuracy measures (for use in the budgeter interface)?
  + For CEM – The accuracy of the number of treated units after matching. Easy to compute because we are just adding Laplacean noise to nT, and the amount of noise doesn’t depend on the data.
  + For difference of means – This accuracy measure would require input of the number of treated units in the dataset. nT could be estimated by the user based solely on the sample size of the data, but it will be an almost arbitrary guess (based solely on the fact that nT is usually less than half of the overall sample size). A user-based nT estimate could also be useful because the user can play with different nT inputs and get a sense of how the accuracy of the difference of means will be affected.
* What do user tests look like for 1-shot, interactive, and automatic CEM?

Clearly, there is much work still to be done until a DP CEM GUI is fully integrated. My project has for the most part made the back-end functionality available, and studied the auo-CEM algorithm. Much more work needs to be done in PSI development and user testing before this can be implemented in an intuitive ways

Thanks to Prof. Salil Vadhan (PI), Dr. Vishesh Karwa (mentor), and Alyssa Hu (co-intern) for the help and support along the way.