HW2: Differential Privacy Foundations

Bhaven Patel

3/12/2019

I worked with Anthony Rentsch and Lipika Ramaswamy on this homework.

My code can be found on my [Github (https://github.com/bhavenp/cs208/blob/master/homework/HW2/HW2_Bhaven_Patel.ipynb)](https://github.com/bhavenp/cs208/blob/master/homework/HW2/HW2_Bhaven_Patel.ipynb).

Problem 1: Mechanisms

I answer parts (a)-(c) for each mechanism below.

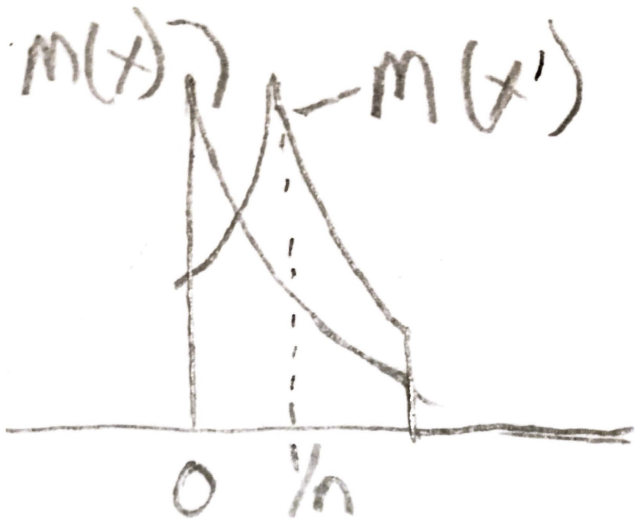
(i) $M(x) = [\bar{x} + Z]_0^1$ for $Z \sim Lap(2/n)$.

(a)

The global sensitivity of $\bar{x}$ is $GS_q = \frac{1}{n}$.

Let us consider the worst case corresponding to this mechanism.

Define $x = \{0, 0, \ldots 0\}$ and and $x' = \{0, \ldots, 0, 1\}$. Then $\bar{x} = 0$ and $\bar{x}' = \frac{1}{n}$. The probability distribution for M(x) and M(x') are bounded between $[0, 1]$ as seen below



To determine whether this mechanism is ϵ-DP, we need to determine whether we can scale up the probability mass of $M(x')$ to cover the probability mass of $M(x)$, since the two distributions overlap.

We can determine if this mechanism is $(\epsilon, 0)$-DP as follows:

$$\begin{aligned}
e^{\epsilon} = \frac{Pr[M(x) = r]}{Pr[M(x') = r} &= \frac{P([\bar{x} + Z]_0^1 = r)}{P([\bar{x}' + Z]_0^1 = r)} \\
&= \frac{\frac{n}{4} exp(-|r - \bar{x}|\frac{n}{2})}{\frac{n}{4} exp(-|r - \bar{x}'|\frac{n}{2})} \\
&= exp\left(\frac{n}{2}|r - \bar{x}'| - |r - \bar{x}|\right) \\
&\leq exp\left(\frac{n}{2}|r - \bar{x}' - r + \bar{x}|\right) \quad \text{(triangle inequality)} \\
&\leq exp\left(\frac{n}{2}| - \frac{1}{n} + 0|\right) \\
&\leq exp\left(\frac{1}{2}\right)
\end{aligned}$$

Thus, we see that this mechanism is $(\epsilon, 0)$-DP when $\epsilon \geq \frac{1}{2}$.

(b)

This mechanism is $(\epsilon, 0)$-DP, so $\delta = 0$.

(c)

If the data domain changes to $[a, b]$ and we want to keep the amount of noise we add at $Lap(\frac{2}{n})$, the minimum value of ϵ changes proportional to $b - a$ because the GS_q becomes $\frac{b-a}{n}$, so ϵ must also grow by a factor of $b - a$ to keep the noise constant.

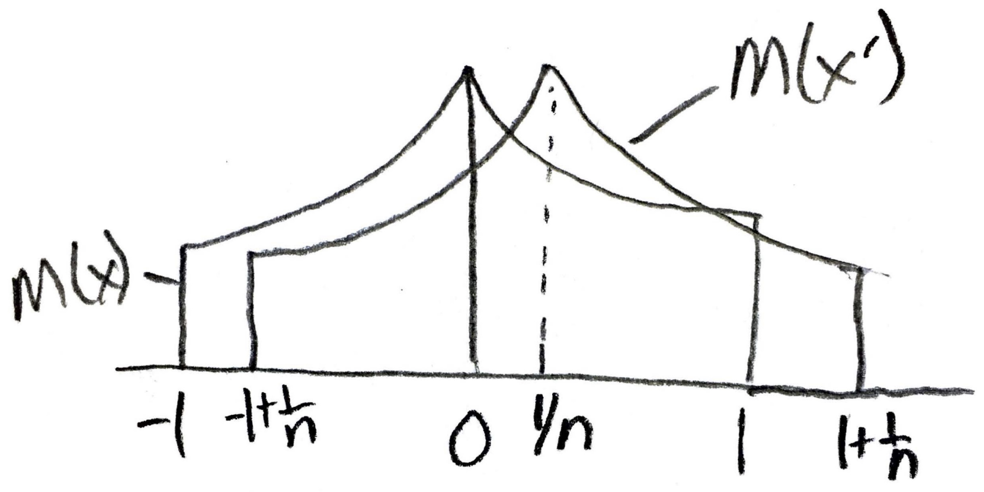
To make ϵ tunable, we just need to change the Laplace noise so that it is $Lap(\frac{GS_q}{\epsilon})$, where $GS_q = \frac{b-a}{n}$.

(ii) $M(x) = \bar{x} + [Z]_{-1}^1$ for $Z \sim Lap(2/n)$

(a)

The global sensitivity of $\bar{x}$ is $GS_q = \frac{1}{n}$.

Define $x = \{0, 0, \ldots 0\}$ and and $x' = \{1, 0, 0, \ldots, 0\}$. Then $\bar{x} = 0$ and $\bar{x}' = \frac{1}{n}$. Since the noise is bounded in $[-1, 1]$, our probability distributions of $M(x)$ and $M(x')$ look like:



As we can see from the picture, $P(M(X) = -1) = c$, where $0 < c < 1$, however $P(M(X') = -1) = 0$. Thus, this mechanism cannot be $(\epsilon, 0)$-DP because $P(M(X) = -1) \nleq e^{\epsilon} P(M(X') = -1)$. **So this mechanism must be (ϵ, δ)-DP.**

(b)

To determine the minimum δ required to make the mechanism (ϵ, δ)-DP, we use the following formula:

$$\delta \geq \max_{x \sim x'} \left[\sum_{y} max\Big[\Pr[M(x) = y] - e^{\epsilon} \cdot \Pr[M(x') = y], 0 \Big] \right]$$

$$\geq \max_{x \sim x'} \left[\int_{-\infty}^{-1+\frac{1}{n}} max\Big[\Pr[M(x) = y] - e^{\epsilon} \cdot \Pr[M(x') = y], 0 \Big] dy \right]$$

Since $P[M(x) = y] \geq 0$ on these bounds and $P[M(x') = y] = 0$, the internal max evaluates to $P[M(x) = y]$:

$$\geq \max_{x \sim x'} \left[\int_{-\infty}^{-1+\frac{1}{n}} \Pr[M(x) = y] dy \right]$$

$$\geq \int_{-\infty}^{-1+\frac{1}{n}} \frac{\exp(\frac{|y-\bar{x}|}{2/n})}{4/n} \partial y$$

$$\geq \frac{n}{4} \int_{-1}^{-1+\frac{1}{n}} \exp\left(\frac{-|y-0|n}{2} \right) \partial y$$
$$\geq \frac{n}{4} \int_{-1}^{-1+\frac{1}{n}} \exp\left(\frac{-yn}{2} \right) \partial y$$
$$\geq \frac{n}{4} \left[\frac{2}{n} e^{yn/2} \right]_{-\infty}^{-1+\frac{1}{n}}$$
$$\geq \frac{n}{4} \cdot \frac{2}{n} \left[e^{(-n+1)/2} - e^{-\infty} \right]$$
$$\geq \frac{1}{2} e^{(-n+1)/2}$$

Thus, this mechanism is $(\epsilon, \frac{1}{2}e^{(-n+1)/2})$-DP.

(c)

If the data domain changes to $[a, b]$, then our δ will need to be $\frac{1}{2}e^{(-n+b-a)/2}$, thus δ must scale by $e^{(\frac{b-a}{n})}$ at minimum to meet the definition of (ϵ, δ)-DP. ϵ is free to take on any value with the corresponding δ. δ cannot exceed 1 because otherwise an outsider could tell with probability 100% whether x or x' was used by the mechanism.

(iii)

$$M(x) = \begin{cases} 1, & \text{w.p. } \bar{x} \\ 0, & \text{w.p. } 1 - \bar{x} \end{cases}$$

(a)

The global sensitivity of $\bar{x}$ is $GS_q = \dfrac{1}{n}$.

Let us consider the worst case corresponding to this mechanism. Let $x = [1, 0, \ldots .0]$ and and $x' = [0, 0, \ldots, 0]$. Then $\bar{x} = \frac{1}{n}$ and $\bar{x}' = 0$. The mechanism is equivalent to a Bernoulli trial with the probability of success set to $\bar{x}$. The probability distributions for $M(x)$ and $M(x')$ are described below

$$M(x) = \begin{cases} 1, & \text{w.p. } \frac{1}{n} \\ 0, & \text{w.p. } 1 - \frac{1}{n} \end{cases}$$

$$M(x') = \begin{cases} 1, & \text{w.p. } 0 \\ 0, & \text{w.p. } 1 \end{cases}$$

Because $P[M(x) = 1] = \dfrac{1}{n}$ and $P[M(x') = 1] = 0$, this mechanism violates definition of $(\epsilon, 0)$-DP because

$$P[M(x) = 1] \nleq e^{\epsilon} P[M(x') = 1].$$

for any value of ϵ. **Thus, this mechanism is not $(\epsilon, 0)$-DP.**

(b)

To determine the minimum δ required to make the mechanism (ϵ, δ)-DP, we use the following formula:

$$\delta \geq \max_{x \sim x'} \left[\sum_y max\Big[\Pr[M(x) = y] - e^{\epsilon} \cdot \Pr[M(x') = y], 0 \Big] \right]$$

Since there are only two possible outputs for the mechanism ($M(x) \in \{0, 1\}$), we can expand the sum as follows:

$$\geq \max_{x \sim x'} \Big[max\Big[\Pr[M(x) = 0] - e^{\epsilon} \cdot \Pr[M(x') = 0], 0 \Big] + max\Big[\Pr[M(x) = 1] - e^{\epsilon} \cdot \Pr[M(x') = 1], 0 \Big]$$

$$\geq \left[max\Big[1 - \frac{1}{n} - e^{\epsilon} \cdot 1, 0\Big] + max\Big[\frac{1}{n} - e^{\epsilon} \cdot 0, 0\Big] \right]$$

$$\geq \left[max \left[\frac{1}{n}, 0\right] \right]$$

$$\geq \frac{1}{n}$$

Thus, this mechanism is $(\epsilon, \frac{1}{n})$-DP.

(c)

Like with (ii), if the data domain changes to $[a, b]$, then our δ will become $\dfrac{b-a}{n}$, which means that the δ will need to be scaled by $b - a$ at minimum so that we meet the definition of (ϵ, δ)-DP. ϵ is free to take on any value with the corresponding δ. Again, δ cannot exceed 1 otherwise an outsider will be able to tell whether dataset x or x' was used by the mechanism.

(iv)

$$M(x) = Y$$

$$f_Y(y) = \begin{cases} \dfrac{e^{-n|y-\bar{x}|/10}}{\int_0^1 e^{-n|y-\bar{x}|/10} dy}, & \text{if} y \in [0,1] \\ 0, & \text{if} y \notin [0,1] \end{cases}$$

(a)

Let us consider the worst case corresponding to this mechanism. Let $x = [0, 0, \ldots 0]$ and and $x' = [0, \ldots, 0, 1]$. Then $\bar{x} = 0$ and $\bar{x}' = \frac{1}{n}$. We can now compute the ratio of the probability distributions for $M(x)$ and $M(x')$ to determine if this mechanism is $(\epsilon, 0)$-DP:

$$\frac{P[M(x) = y]}{P[M(x') = y]} = \frac{\dfrac{e^{-n|y-\bar{x}|/10}}{\int_0^1 e^{-n|y-\bar{x}|/10} dy}}{\dfrac{e^{-n|y-\bar{x}'|/10}}{\int_0^1 e^{-n|y-\bar{x}'|/10} dy}}$$

$$= \frac{e^{-n|y-\bar{x}|/10}}{e^{-n|y-\bar{x}'|/10}} * \frac{\int_0^1 e^{-n|y-\bar{x}'|/10} dy}{\int_0^1 e^{-n|y-\bar{x}|/10} dy}$$

We will now treat these two terms independently. First, I will find an upper bound for the left fraction:

$$\frac{e^{-n|y-\bar{x}|/10}}{e^{-n|y-\bar{x}'|/10}} = e^{\frac{1}{10}\left(-n|y-\bar{x}|+n|y-\bar{x}'|\right)}$$

$$= e^{\frac{n}{10}\left(|y-\bar{x}'|-|y-\bar{x}|\right)}$$

$$\leq e^{\frac{n}{10}|\bar{x}-\bar{x}'|} \quad \text{using the triangle inequality}$$

$$\leq e^{\frac{n}{10}|0-\frac{1}{n}|}$$

$$\leq e^{\frac{n}{10}\left(\frac{1}{n}\right)}$$

$$\leq e^{\frac{1}{10}}$$

So we find $\dfrac{e^{-n|y-\bar{x}|/10}}{e^{-n|y-\bar{x}'|/10}} \leq e^{\frac{1}{10}}$.

Now, we can find an upper bound for the right hand side of fraction:

$$\frac{\int_0^1 e^{-n|y-\bar{x}'|/10} dy}{\int_0^1 e^{-n|y-\bar{x}|/10} dy} = \frac{\int_0^1 e^{-n|y-\frac{1}{n}|/10} dy}{\int_0^1 e^{-n|y-0|/10} dy}$$

$$= \frac{\int_0^{1/n} e^{-\frac{n}{10}|y-\frac{1}{n}|} dy + \int_{\frac{1}{n}}^1 e^{-\frac{n}{10}|y-\frac{1}{n}|} dy}{\int_0^{1/n} e^{-\frac{n}{10}y} dy + \int_{\frac{1}{n}}^1 e^{-\frac{n}{10}y} dy}$$

$$= \frac{\int_0^{1/n} e^{-\frac{n}{10}y+\frac{1}{10}} dy + \int_{\frac{1}{n}}^1 e^{-\frac{n}{10}y+\frac{1}{10}} dy}{\int_0^{1/n} e^{-\frac{n}{10}y} dy + \int_{\frac{1}{n}}^1 e^{-\frac{n}{10}y} dy}$$

$$= \frac{e^{\left(\frac{1}{10}\right)} \left(\int_0^{1/n} e^{-\frac{n}{10}y} dy + \int_{\frac{1}{n}}^1 e^{-\frac{n}{10}y} dy\right)}{\int_0^{1/n} e^{-\frac{n}{10}y} dy + \int_{\frac{1}{n}}^1 e^{-\frac{n}{10}y} dy}$$

$$= e^{\left(\frac{1}{10}\right)}$$

Thus, we find that

$$\frac{e^{-n|y-\bar{x}|/10}}{e^{-n|y-\bar{x}'|/10}} * \frac{\int_0^1 e^{-n|y-\bar{x}'|/10} dy}{\int_0^1 e^{-n|y-\bar{x}|/10} dy} \leq e^{\left(\frac{1}{10}\right)} e^{\left(\frac{1}{10}\right)}$$

$$\leq e^{\left(\frac{1}{5}\right)}$$

Thus, we see that this mechanism is $(\epsilon, 0)$-DP when $\epsilon \geq \frac{1}{5}$.

(b)
This mechanism is $(\epsilon, 0)$-DP, so $\delta = 0$.

(c)
If the data domain changes to $[a, b]$, the minimum value of ϵ becomes $\dfrac{b-a}{5}$, so ϵ changes proportional to $b - a$ because the GS_q becomes $\dfrac{b-a}{n}$.

Problem 1

(d) One of the $(\epsilon, 0)$-DP mechanisms (either *i* or *iv*) would be best for releasing the mean, since they will be differentially private 100% of the time. I would choose mechanism *i* because less noise needs to be added initially (minimum $\epsilon = 0.5$) for this mechanism to be differentially private, while mechanism *(iv)* requires an $\epsilon = 0.2$. Thus, the utility from the answer for mechanism *i* would be higher than the utility from the answer for mechanism *iv*.

Problem 2: Evaluating DP Algorithms with Synthetic Data

In [1]:

```
rm(list=ls())        # Remove any objects in memory
```

(a)
Create a data generating function that samples from the Poisson distribution.

In [2]:

```
# Random draws from Poisson distribution
#
# mean- numeric, mean of the distribution
# size integer, number of draws
#
# return Random draws from Laplace distribution
# example:
#
# sample_poisson(num_draws=1000)

sample_poisson <- function(mean=10, num_draws=1){
    data = rpois(n = num_draws, lambda = mean);
    return(data);
}
```

(b)

I picked the first mechanism from Problem 1: $M(x) = [\bar{x} + Z]_0^1$, $Z \sim Lap\left(\frac{GS}{\epsilon}\right)$

In [3]:

```
# Sign function
#
# Function to determine what the sign of the passed values should be.
#
# x numeric, value or vector or values
# return The sign of passed values
# example:
#
# sgn(rnorm(10))
# Taken from James Honaker's histogramRelease.r
sgn <- function(x) {
    return(ifelse(x < 0, -1, 1))
}

# Random draw from Laplace distribution
#
# mu numeric, center of the distribution
# b numeric, spread
# size integer, number of draws
#
# return Random draws from Laplace distribution
# example:
#
# rlap(size=1000)
# Taken from James Honaker's histogramRelease.r
rlap = function(mu=0, b=1, size=1) {
    p <- runif(size) - 0.5
    draws <- mu - b * sgn(p) * log(1 - 2 * abs(p))
    return(draws)
}

## Clip a variable to a range. Taken from James Honaker's histogramRelease.r
clip <- function(x, lower, upper){
    x.clipped <- x
    x.clipped[x.clipped<lower] <- lower
    x.clipped[x.clipped>upper] <- upper
    return(x.clipped)
}

## Differentially private mean release. Taken from James Honaker's
laplaceMeanReleaseFull.r
meanRelease <- function(x, lower=0, upper, epsilon){
    n <- length(x); #get number of data points/observations

    sensitivity <- (upper - lower)/n; #calculate GS
    scale <- sensitivity / epsilon; #calculate scale for laplace noise

    x.clipped <- clip(x, lower, upper);
    sensitiveValue <- mean(x.clipped);
    DPrelease <- sensitiveValue + rlap(mu=0, b=scale, size=1);
    DPrelease <- clip(DPrelease, lower, upper); #clip the DP-mean if it
lower/greater than the desired bounds
```

```
    return(list(release=DPrelease, true_clipped=sensitiveValue ));
}
```

(c)

For n=200 and ϵ=0.5, plot the root mean squared error as a function of the upper bound b

In [4]:

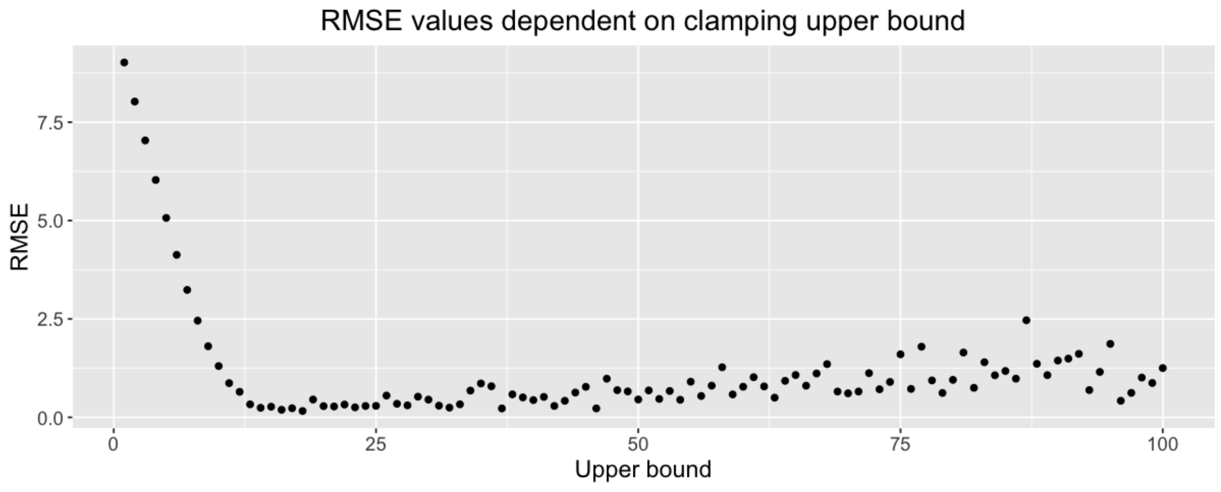
```
#set parameters
n = 200;
epsilon = 0.5;
num_sims = 10; #number of simulations to run for each upper bound
```

In [5]:

```
#set the seed
set.seed(24);

b_vals = seq(from = 1, to=100); #create sequence of upper bounds

datasets <- matrix(NA, nrow=n, ncol=num_sims);
#generate 'num_sims' different datasets from Poisson distribution
for(i in 1:num_sims){
    datasets[,i] = sample_poisson(num_draws = n); #put i-th dataset in column i
}
dataset_means = colMeans(datasets); #calculate mean for each dataset

#create matrix to hold upper bound and rmse for mean DP-releases, average mean
rmse_vals = matrix(NA, nrow=length(b_vals), ncol=3);
for(b in b_vals){
    dp_means = c(); #vector to hold DP-released means for each simulation
    for(i in 1:num_sims){
        results = meanRelease(datasets[,i], lower = 0, upper = b, epsilon =
epsilon); #get mean release
        dp_means <- c(dp_means, results$release);
    }

    #calculate RMSE
    rmse = ( sum( (dp_means - dataset_means)**2 ) / length(dp_means)) ** 0.5;
    rmse_vals[b,] <- c(b, rmse, mean(dp_means));
}
```

In [6]:

```
library(ggplot2); #import library for plotting
library(grid);

#### Plot results ####
final_results <- as.data.frame(rmse_vals);
colnames(final_results) <- c("Upper_bound", "RMSE", "Average true mean");
f_size = 16;
fifty = 0.5;
# Plot average RMSE of reconstruction against noise input
p_rmse <- ggplot(data = final_results, aes(x=final_results$Upper_bound,
y=final_results$RMSE)) + geom_point();
p_rmse <- p_rmse + labs(title="RMSE values dependent on clamping upper bound",
x="Upper bound", y = "RMSE") + theme(plot.title = element_text(hjust=0.5), text =
element_text(size=f_size-2));
options(repr.plot.width=10, repr.plot.height=4); #set plot dimensions
p_rmse #show plot
```



In [7]:

```
print(final_results[final_results$RMSE == min(final_results$RMSE), ]);
```

```
   Upper_bound     RMSE Average true mean
18          18 0.161492          9.954823
```

The optimal value b^* is 18. From the graph, it looks like values from 16 to 18 all have comparable RMSE values and could be used as good upper bounds.

(d)
If we generate an optimal value b^* based on bootstrapped samples from our dataset x, then we may leak information about outliers in the our dataset. This would occur because some of the bootstraps will contain the outliers, and their RMSE's will be minimized when the upper bound for clipping is near or greater than the value of the outlier. Thus by looking at the graph of average RMSE versus upper bound for the bootstrapped samples, it is possible to infer what the values of the outliers could be. This would not be differentially private because outliers in the dataset could be identified. Additionally by choosing the b^* based on the dataset, we would most likely overfit to the dataset.

(e)
If we have some idea of what the data universe looks like (we have some prior), we could generate many alternative/synthetic datasets and calculate the best upper bound based on those synthetic datasets. This would prevent us from leaking any private information from our own dataset. If we don't have this prior, we could build a DP histogram and build multiple synthetic datasets from this DP histogram; these synthetic datasets would then be used to calculate the best upper bound.

Problem 3: Regression

(a)

To produce a differentially-private release of $\hat{\alpha}$, I will create differentially private releases of the covariance of x and y (S_{xy}), the variance of x (S_{xx}), the mean of y ($\bar{y}$), and the mean of x ($\bar{x}$). If our privacy budget is ϵ, each of these releases will take a fraction of the privacy budget equal to $\frac{\epsilon}{4}$, and due to the composition rule, the total privacy loss will be at most ϵ.
$\hat{\beta}$ and $\hat{\alpha}$ can be calculated from the differentially private releases of S_{xy}, S_{xx}, $\bar{y}$, and $\bar{x}$ as follows:

$$\hat{\beta} = \frac{S_{xy}}{S_{xx}}$$

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

Below is my augmentation to the regression release code so that I also give a differentially private release of $\hat{\alpha}$.

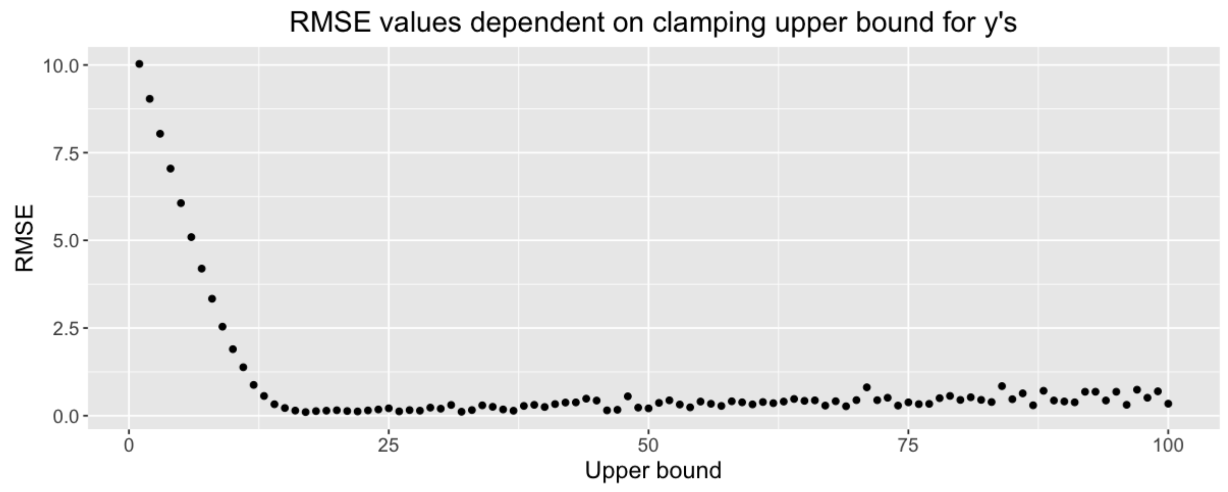
In [5]:

```
## Differentially private regression slope release. Modified from
laplaceRegressionRelease.r by James Honaker.
regressionRelease <- function(y, x, ylower, yupper, xlower, xupper,
epsilon_parts){
    x.clipped <- clip(x, xlower, xupper)
    y.clipped <- clip(y, ylower, yupper)

    n <- length(x)
    sens.Sxy <- ((xupper-xlower)*(yupper-ylower))
    sens.Sxx  <- ((xupper-xlower)^2)

    scale.Sxy <- sens.Sxy / (epsilon_parts$Sxy);
    scale.Sxx <- sens.Sxx / (epsilon_parts$Sxx);

    sensitiveValue <- sum((x.clipped - mean(x.clipped))*(y.clipped -
mean(y.clipped))) / sum((x.clipped -

             mean(x.clipped))^2);

    #get DP-releases of Sxy and Sxx to calculate beta-hat
    release.Sxy <- sum((x.clipped - mean(x.clipped))*(y.clipped -
mean(y.clipped)))  + rlap(mu=0, b=scale.Sxy, size=1);
    release.Sxx <- sum((x.clipped - mean(x.clipped))^2)            + rlap(mu=0,
b=scale.Sxx, size=1);
    #get DP-releases of x-bar and y-bar to calculate alpha-hat
    release.xbar <- meanRelease(x, lower = xlower, upper = xupper, epsilon =
epsilon_parts$xbar)$release;
    release.ybar <- meanRelease(y, lower = ylower, upper = yupper, epsilon =
epsilon_parts$ybar)$release;

    #calculate beta-hat and alpha-hat
    postprocess.beta <- release.Sxy/release.Sxx;
    postprocess.alpha <- release.ybar - postprocess.beta * release.xbar;

    true.alpha = mean(y.clipped) - sensitiveValue * mean(x.clipped); #calculate
the true alpha of the clipped values
    return(list(dp_beta=postprocess.beta, true_beta=sensitiveValue,
dp_alpha=postprocess.alpha,
                true_alpha=true.alpha))
}
```

First, I want to find a good upper bound for clamping the y's.

In [9]:

```
#set the seed
set.seed(24);

n = 200;
num_sims = 10; #number of simulations to run for each upper bound
alpha = beta = sigma = epsilon = 1;

b_vals = seq(from = 1, to=100); #create sequence of upper bounds

datasets <- matrix(NA, nrow=n, ncol=num_sims);
#generate 'num_sims' different datasets from Poisson distribution
for(i in 1:num_sims){
    x_data = sample_poisson(num_draws = n);
    datasets[,i] = alpha*x_data + beta + rnorm(n=n, mean=0, sd=sigma); #put i-th
dataset in column i
}
dataset_means = colMeans(datasets); #calculate mean for each dataset

#create matrix to hold upper bound and rmse for mean DP-releases, average mean
rmse_vals = matrix(NA, nrow=length(b_vals), ncol=3);
for(b in b_vals){
    dp_means = c(); #vector to hold DP-released means for each simulation
    for(i in 1:num_sims){
        results = meanRelease(datasets[,i], lower = 0, upper = b, epsilon =
epsilon); #get mean release
        dp_means <- c(dp_means, results$release);
    }

    #calculate RMSE
    rmse = ( sum( (dp_means - dataset_means)**2 ) / length(dp_means)) ** 0.5;
    rmse_vals[b,] <- c(b, rmse, mean(dp_means));
}

#### Plot results ####
library(ggplot2)
final_results <- as.data.frame(rmse_vals);
colnames(final_results) <- c("Upper_bound", "RMSE", "Average true mean");
f_size = 16;
fifty = 0.5;
# Plot average RMSE of reconstruction against noise input
p_rmse <- ggplot(data = final_results, aes(x=final_results$Upper_bound,
y=final_results$RMSE)) + geom_point();
p_rmse <- p_rmse + labs(title="RMSE values dependent on clamping upper bound for
y's", x="Upper bound", y = "RMSE") + theme(plot.title = element_text(hjust=0.5),
text = element_text(size=f_size-2));
options(repr.plot.width=10, repr.plot.height=4); #set plot dimensions
p_rmse #show plot
```



In [10]:

```
print(final_results[final_results$RMSE == min(final_results$RMSE), ]);
```

```
   Upper_bound      RMSE Average true mean
17          17 0.1037066          10.96932
```

I will use an upper bound of 18 for x and an upper bound of 17 for y.

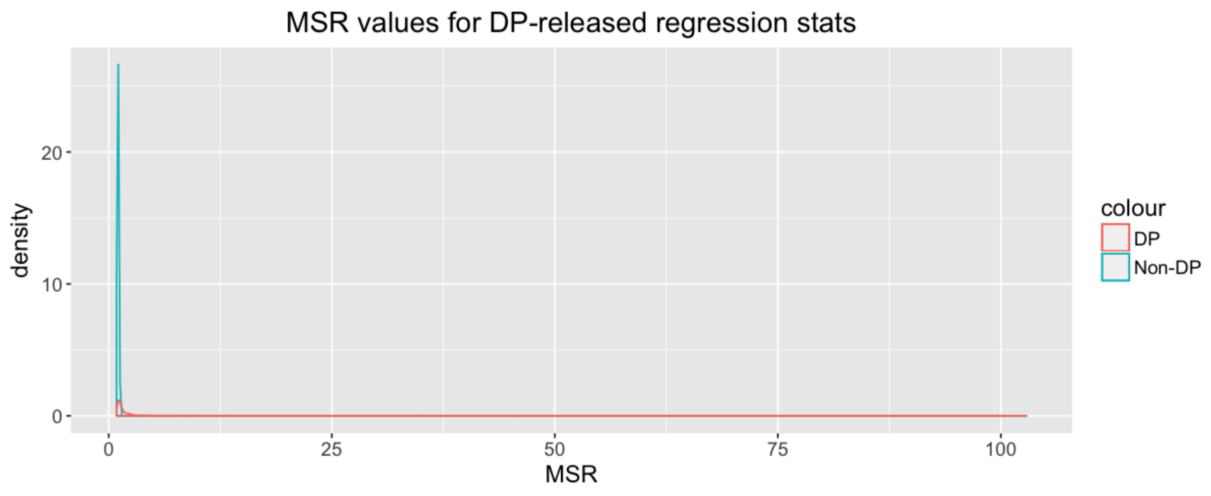
(b)

In [8]:

```
#set the seed
set.seed(24);

#set parameters
y_lower = 0;
y_upper = 17;
x_lower = 0;
x_upper = 18;


n = 1000; #size of the dataset
num_sims = 1000; #number of Monte Carlo simulations to run
alpha = beta = sigma = epsilon = 1;

#define partitions for epsilon, which should all be epsilon/4
epsilon_parts = list(Sxy=epsilon/4, Sxx=epsilon/4, xbar=epsilon/4,
ybar=epsilon/4);

#create matrix to hold upper bound and rmse for mean DP-releases, average mean
rmse_vals = matrix(NA, nrow=num_sims, ncol=3);
for(i in 1:num_sims){
    #generate x-data and y-data
    x_data = sample_poisson(num_draws = n);
    y_data = beta*x_data + alpha + rnorm(n=n, mean=0, sd=sigma);
    #get regression release beta and alpha
    reg_release <- regressionRelease(y_data, x_data, ylower=y_lower,
yupper=y_upper,
                                     xlower=x_lower, xupper=x_upper,
epsilon_parts=epsilon_parts);

    #calculate y for each x point in dataset using regression release statistics
    y_dp <- reg_release$dp_beta * x_data + reg_release$dp_alpha;
    y_nondp <- reg_release$true_beta * x_data + reg_release$true_alpha; #calc y
using non-dp stats
    #calculate RMSE for the DP release stats
    rmse_dp = sum( (y_dp - y_data)**2 ) / length(y_data);
    #calculate RMSE for the non-DP release stats
    rmse_nondp = sum( (y_nondp - y_data)**2 ) / length(y_data);

    rmse_vals[i,] <- c(i, rmse_dp, rmse_nondp); #put RMSE values into matrix
}
```

In [10]:

```
#### Plot results ####
library(ggplot2)
final_results <- as.data.frame(rmse_vals);
colnames(final_results) <- c("Simulation", "MSE_DP", "MSE_nonDP");
f_size = 16;

# Plot average RMSE of reconstruction against noise input #
#geom_density
p_rmse <- ggplot(data = final_results) +
geom_density(aes(x=final_results$MSE_nonDP, color='Non-DP')) +
geom_density(aes(x=final_results$MSE_DP, color='DP' ));
p_rmse <- p_rmse + labs(title="MSR values for DP-released regression stats",
x="MSR", y="density") + theme(plot.title = element_text(hjust=0.5), text =
element_text(size=f_size-2));
options(repr.plot.width=10, repr.plot.height=4); #set plot dimensions

p_rmse #show plot
```



The distribution of mean-squared residuals for the differentially private released regression stats is right-skewed because some of the stats must receive a large amount of noise from the Laplace distribution resulting in very high residuals for some DP-releases.

(c)

In [23]:

```
#create a vector with values from 0.1 to 0.9
eps_to_test <- seq(0.1,0.9,0.1);

#create data frame of all combination of values from vectors
eps_combos_to_test <- as.data.frame(expand.grid(list(a=eps_to_test,
b=eps_to_test, c=eps_to_test, d=eps_to_test) ));
```

In [37]:

```
#normalize values in each row so they add up to 1
eps_combos_to_test_norm = eps_combos_to_test / rowSums(eps_combos_to_test)
# eps_combos_to_test_norm

#remove the duplicates
dups = which(duplicated(eps_combos_to_test_norm));
eps_combos_to_test_norm = eps_combos_to_test_norm[-dups, ];
print(paste("I need to test", nrow(eps_combos_to_test_norm), "combinations") );
```

```
[1] "I need to test 6223 combinations"
```

In [92]:

```
set.seed(24); #set the seed for reproducible results

#get a single dataset to test on
x_data = sample_poisson(num_draws = n);
y_data = alpha*x_data + beta + rnorm(n=n, mean=0, sd=sigma);

num_sim = 10;

#create matrix to hold epsilon partitions and MSR for DP-releases
partition_res = matrix(NA, nrow=nrow(eps_combos_to_test_norm), ncol=5);

#go through each epsilon partition I calculated
for(i in 1:nrow(eps_combos_to_test_norm)){
    epsilon_parts = list(Sxy=eps_combos_to_test_norm[i,1],
Sxx=eps_combos_to_test_norm[i,2],
                         xbar=eps_combos_to_test_norm[i,3],
ybar=eps_combos_to_test_norm[i,4]);
    msr_vec = c();
    for(j in 1:num_sim){
        #get regression release beta and alpha
        reg_release <- regressionRelease(y_data, x_data, ylower=y_lower,
yupper=y_upper,
                                     xlower=x_lower, xupper=x_upper,
epsilon_parts=epsilon_parts);

        #calculate y for each x point in dataset using regression release
statistics
        y_dp <- reg_release$dp_beta * x_data + reg_release$dp_alpha;

        #calculate RMSE for the DP release stats
        msr_dp = sum( (y_dp - y_data)**2 ) / length(y_data);
        msr_vec = c(msr_vec, msr_dp);
    }

    partition_res[i,] <- c( as.numeric(eps_combos_to_test_norm[i, ]),
mean(msr_dp) ); #put partition values and mean of MSR values with these partition
values into the matrix
}

#convert the results to a data frame
partition_res_df = as.data.frame(partition_res);
```

In [93]:

```
#give columns names
colnames(partition_res_df) <- c("Sxy_eps", "Sxx_eps", "xbar_eps", "ybar_eps",
"MSR");
#sort based on MSR
partition_res_df <- partition_res_df[order(partition_res_df$MSR), ];
partition_res_df[1:5, ]
```

	Sxy_eps	Sxx_eps	xbar_eps	ybar_eps	MSR
2692	0.1666667	0.11111111	0.5000000	0.22222222	0.9843554
3318	0.3103448	0.27586207	0.2413793	0.17241379	0.9844241
5853	0.3333333	0.08333333	0.2083333	0.37500000	0.9844889
637	0.2916667	0.33333333	0.3333333	0.04166667	0.9845111
6217	0.2571429	0.22857143	0.2571429	0.25714286	0.9845811

In [94]:

```
median_msr <- median(final_results$MSE_DP);
print(paste("Median MSR for DP release with equal epsilion partitions from (b):",
median_msr) );
print("Mean MSR for DP release with for best parition:");
partition_res_df[1,]
```

```
[1] "Median MSR for DP release with equal epsilion partitions from
(b): 1.24757505362371"
[1] "Mean MSR for DP release with for best parition:"
```

	Sxy_eps	Sxx_eps	xbar_eps	ybar_eps	MSR
2692	0.1666667	0.1111111	0.5	0.2222222	0.9843554

To determine the best partition of ϵ for the four regression stats releases S_{xy}, S_{xx}, $\bar{x}$, and $\bar{y}$, I calculated DP-releases for a single dataset 10 times for each partition of ϵ. I decided to to do a DP-release on the same dataset 10 times so that the only noise in the results was derived from the Laplace noise added to the regression stats each time they were calculated. I tested 6223 different partitions of ϵ for the four regression stats and found that the above partition gave the best mean-squared residual on the single dataset. The mean-squared residual for my best partition gave better utility than the equal partition that I tested in part (b). Several other partitions of ϵ gave almost identical results to my best partition, suggesting there may be several good choices. From the top five scoring partitions, it does not seem like giving a higher or lower partition to any of the DP-released stats is better; the values for all them seem to jump around quite a bit. Another strategy for choosing the best partition would be to create 10 different datasets, calculate the mean-squared error for a given ϵ partition for each dataset, and compare the averages of the mean-squared error of all the different partitions.

Problem 4: DP vs. Reconstruction Attacks

$$E[\#\{i \in [n] : A(M(X))_i = X_i\}/n]$$

represents the expected fraction of bits the attacker can reconstruct. This can be written as

$$= \frac{1}{n} E[\#\{i \in [n] : A(M(X))_i = X_i\}]$$
$$= \frac{1}{n} \sum_{i=1}^{n} P[A(M(X_i, X_{-i})) = X_i]$$

where X_i is the row the attacker is trying to reconstruct and X_{-i} is all elements in the dataset except X_i. Using the definition of (ϵ, δ)-DP and the rule of post-processing, we can write

$$\leq \frac{1}{n} e^{\epsilon} \sum_{i=1}^{n} P[A(M(0, X_{-i})) = X_i] + \delta$$

We change $X_i \rightarrow 0$ because the attacker should not be able to learn anything about one individual from a differentially private release. Anything the attacker learns couuld have been learned by the attacker even without the query.

Now, the best strategy for the attacker to employ is to assume that everyone in the dataset has a sensitive bit equal to the bit with the highest probability. We have a binary case where $X_i \in \{0, 1\}$, and the probabilities of each case can be written as $P[A(M(0, X_{-i})) = 0] = (1 - p)$ and $P[A(M(0, X_{-i})) = 1] = (p$ respectively. If we take the max of the two probabilities, our expression can be written as

$$\leq \frac{1}{n} e^{\epsilon} \sum_{i=1}^{n} max\,(P[A(M(0, X_{-i})) = 0], P[A(M(0, X_{-i})) = 1]) + \delta$$
$$\leq \frac{1}{n} e^{\epsilon} \sum_{i=1}^{n} max\,(1 - p, p) + \delta$$
$$\leq \frac{1}{n} e^{\epsilon} (n) max\,(1 - p, p) + \delta$$
$$\leq e^{\epsilon} max\,(1 - p, p) + \delta$$

Since $e^{\epsilon} \geq 1$, the adversary can reconstruct a fraction of bits slightly larger than just predicting the $max\,(1 - p, p)$.

Problem 5: Final Project Next Step

This will be submitted separately on Gradescope.