cs208 HW 2

Anthony Rentsch

3/12/2019

Note: I include the code I wrote directly in line for some questions, such as for 2 *(a)* and *(b)* which are purely implementation problems. For other questions, my code can be found in the Appendix and on Github.

Question 1

(a) Both (i) and (iv) are (ϵ, 0)-differentially private. I demonstrate that below using two neighboring datasets, $x = [0, 0, ..., 0]$ and $x' = [0, 0, ..., 1]$.

(i)

$$\frac{P([\bar{x} + Z]_0^1 = r)}{P([\bar{x'} + Z]_0^1 = r)}$$

$$\frac{\frac{n}{4} exp(-\frac{n}{2}|r - \bar{x}|)}{\frac{n}{4} exp(-\frac{n}{2}|r - \bar{x'}|)}$$

$$exp(\frac{n}{2}|r - \bar{x'}| - |r - \bar{x}|)$$

By the triangle inequality, this expression is less than or equal to

$$exp(\frac{n}{2}|r - \bar{x'} - r + \bar{x}|) = exp(\frac{n}{2}|\bar{x} - \bar{x'}|)$$

Since $|\bar{x} - \bar{x'}|$ is just the global sensitivity, we get that

$$exp(\frac{n}{2}|\bar{x} - \bar{x'}|) = exp(\frac{n}{2}\frac{1}{n}) = exp(\frac{1}{2})$$

Thus, $M(x)$ is $(\epsilon, 0)$-differentially private for $\epsilon \geq 0.5$.

(iv)

$$\frac{exp(\frac{-n}{10}|y - \bar{x}|)}{exp(\frac{-n}{10}|y - \bar{x'}|)} * \frac{\int_0^1 exp(\frac{-n}{10}|z - \bar{x}|)dz}{\int_0^1 exp(\frac{-n}{10}|z - \bar{x'}|)dz}$$

I'll evaluate this term by term. First, the left term:

$$exp(\frac{n}{10}(-|y - \bar{x}| + |y - \bar{x'}|))$$

$$exp(\frac{n}{10}(-|y| + |y - \frac{1}{n}|))$$

By the triangle inequality, this is less than or equal to

$$exp(\frac{n}{10}(y - \frac{1}{n} - y)) = exp(\frac{1}{10})$$

Now, for the right term:

$$\frac{\int_0^1 exp(\frac{-n}{10}|z - \frac{1}{n}|)dz}{\int_0^1 exp(\frac{-nz}{10})dz}$$

$$\frac{\int_0^{\frac{1}{n}} exp(\frac{-n}{10}(z - \frac{1}{n}))dz + \int_{\frac{1}{n}}^1 exp(\frac{-n}{10}(z - \frac{1}{n}))dz}{\int_0^{\frac{1}{n}} exp(\frac{-nz}{10})dz + \int_{\frac{1}{n}}^1 exp(\frac{-nz}{10})dz}$$

$$\frac{exp(\frac{1}{10}) \int_0^{\frac{1}{n}} exp(\frac{-nz}{10})dz + exp(\frac{1}{10}) \int_{\frac{1}{n}}^1 exp(\frac{-nz}{10})dz}{\int_0^{\frac{1}{n}} exp(\frac{-nz}{10})dz + \int_{\frac{1}{n}}^1 exp(\frac{-nz}{10})dz}$$

This reduces to $exp(\frac{1}{10})$. Putting the two terms together, we have

$$exp(\frac{1}{10}) * exp(\frac{1}{10}) = exp(\frac{1}{5})$$

Thus, this mechanism is $(\epsilon, 0)$-differentially private for $\epsilon \geq 0.2$.

(b) Mechanisms (ii) and (iii) are not $(\epsilon, 0)$-differentially private. Below I'll provide a counterexample that demonstrates this and find a minimum value of δ for which they are (ϵ, δ)-differentially private.

(ii) Consider $x = [0, 0, ..., 0]$ and $x' = [0, 0, ..., 1]$. Now, $P(M(x) = -1) \geq 0$ while $P(M(x') = -1) = 0$. This violates $P(M(x) = -1) \leq exp(\epsilon)P(M(x') = -1)$, so this mechanism is not $(\epsilon, 0)$-differentially private. Now let's consider the minimum value of δ for which it is (ϵ, δ)-differentially private.

$$\delta \geq max_{x\ x'}[\int_y max(P(M(x) = y) - exp(\epsilon)P(M(x') = y), 0)]$$

Since y will be bounded on [-1, 2] here, this is the same as

$$max[\int_{-1}^2 P(M(x) = y) - \exp(\epsilon)P(M(x') = y), 0]$$

Consider the worst-case scenario I defined above, where $x = [0, ..., 0]$ and $x' = [0, ..., 0, 1]$. The expression inside the integral will only be ≥ 0 for $y \in [-1, -1 + \frac{1}{n}]$ because $P(M(x') = y) = 0$ here.

$$\int_{-1}^{-1+\frac{1}{n}} P(M(x) = y)$$

$$\int_{-1}^{-1+\frac{1}{n}} P(\bar{x} + Z = y)$$

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

$$\int_{-1}^{-1+\frac{1}{n}} \frac{n}{4} exp(\frac{-n(y-\bar{x})}{2})$$

$$\frac{n}{4} exp(\frac{-n\bar{x}}{2}) \int_{-1}^{-1+\frac{1}{n}} exp(\frac{-ny}{2})$$

We know that $\bar{x} = 0$ and after integrating we are left with

$$\frac{1}{2}[exp(\frac{ny}{2})]_{-1}^{-1+\frac{1}{n}}$$

$$\frac{1}{2}[exp(\frac{1}{2})exp(\frac{-n}{2}) - exp(\frac{-n}{2})]$$

$$\frac{1}{2}exp(\frac{-n}{2})[exp(\frac{1}{2}) - 1]$$

Thus, this mechanism is (ϵ, δ)-differentially private for $\delta \geq \frac{1}{2}exp(\frac{-n}{2})[exp(\frac{1}{2}) - 1]$.

(iii) Consider $x = [0, 0, ..., 1]$ and $x' = [0, 0, ..., 0]$. Now, $P(M(x) = 1) = \frac{1}{n}$ while $P(M(x') = 1) = 0$. This clearly violates $P(M(x) = 1) \leq exp(\epsilon)P(M(x') = 1)$, so this mechanism is not $(\epsilon, 0)$-differentially private. Now let's consider the minimum value of δ for which it is (ϵ, δ)-differentially private.

$$\delta \geq max_{x\ x'}[\Sigma_y max(P(M(x) = y) - exp(\epsilon)P(M(x') = y), 0)]$$

$$\Sigma_{y\in[0,1]} max[(P(M(x) = y) - exp(\epsilon)P(M(x') = y), 0]$$
$$max[P(M(x) = 1) - exp(\epsilon)P(M(x') = 1), 0] + max[P(M(x) = 0) - exp(\epsilon)P(M(x') = 0), 0]$$

$$max[\frac{1}{n} - exp(\epsilon) * 0, 0] + max[1 - \frac{1}{n} - exp(\epsilon) * 1, 0] = \frac{1}{n} + 0 = \frac{1}{n}$$

Thus, this mechanism is (ϵ, δ)-differentially-private for $\delta \geq \frac{1}{n}$.

(c)

(d) I guess (i) -we understand laplace mechanism well -clamping to protect outliers?

Question 2

(a)

```
poissonDGP <- function(n){ return(rpois(n, lambda=10)) }
```

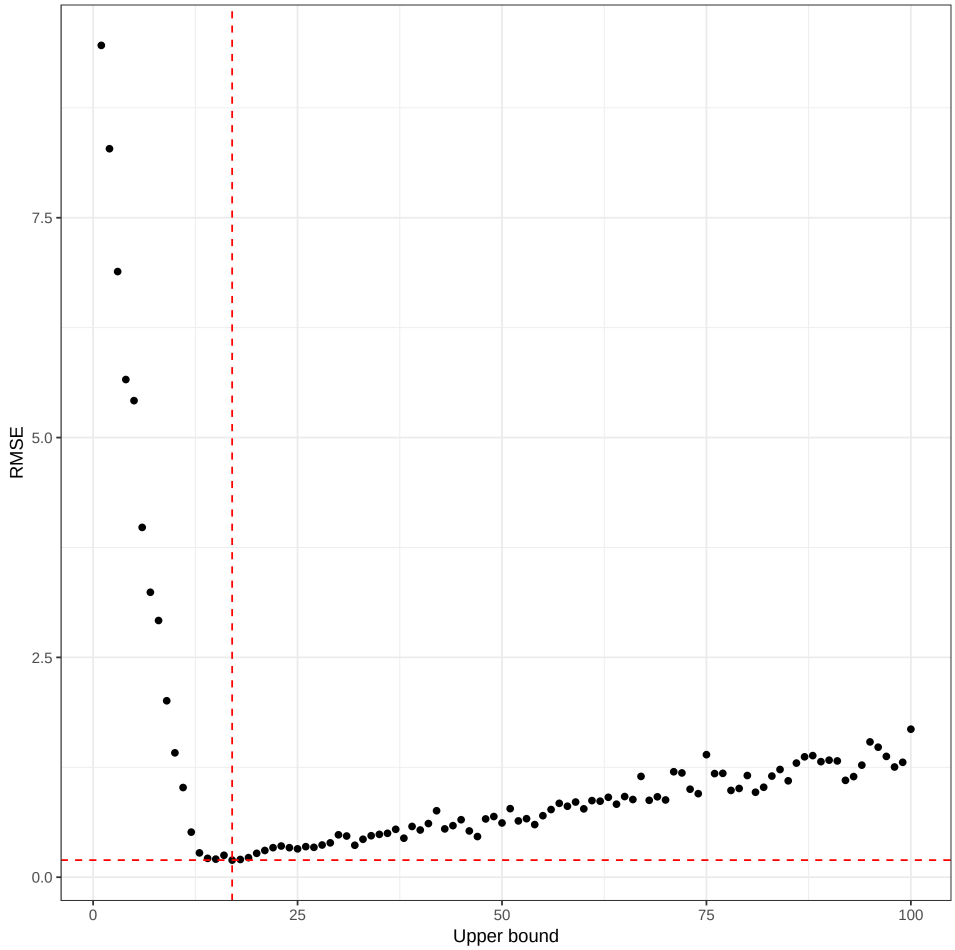
(b) I use the first mechanism from Question 1 to answer this question. From here on out I'll refer to it as the clamped Laplace mean release mechanism.

```
sgn <- function(x) {     # function borrowed from class
  return(ifelse(x < 0, -1, 1))
}

rlap = function(mu=0, b=1, size=1) {     # function borrowed from class
  p <- runif(size) - 0.5
```

```
  draws <- mu - b * sgn(p) * log(1 - 2 * abs(p))
  return(draws)
}

clip <- function(x, lower, upper){     # function borrowed from class
  x.clipped <- x
  x.clipped[x.clipped<lower] <- lower
  x.clipped[x.clipped>upper] <- upper
  return(x.clipped)
}

laplaceClampMeanRelease <- function(x, epsilon, a=0, b=1){
  n <- length(x)
  sensitivity <- (a - b)/n
  scale <- sensitivity / epsilon

  x.clipped <- clip(x, a, b)
  clipped.mean <- mean(x.clipped)
  noisy.mean <- clipped.mean + rlap(mu=0, b=scale, size=1)
  release.mean <- clip(noisy.mean, a, b)
  true.mean <- mean(x)

  return(list(release=release.mean, true=true.mean))
}
```

(c) I run 100 Monte Carlo simulations per upper bound value. In each simulation I draw 200 samples randomly drawn from a Poisson distribution and compute a noisy mean using the clamped Laplace mean mechanism. Then, I compare the root mean squared error between the true and noisy means for each upper bound value and plot these for every upper bound value. This is shown in the plot below.

Based on my analysis, the optimal value for the upper bound b - i.e., the one that minimizes RMSE - is 19. I will use that value for the remiander of this problem and for Question 3.



(d) -protecting outliers -why doesn't bootstrap do this

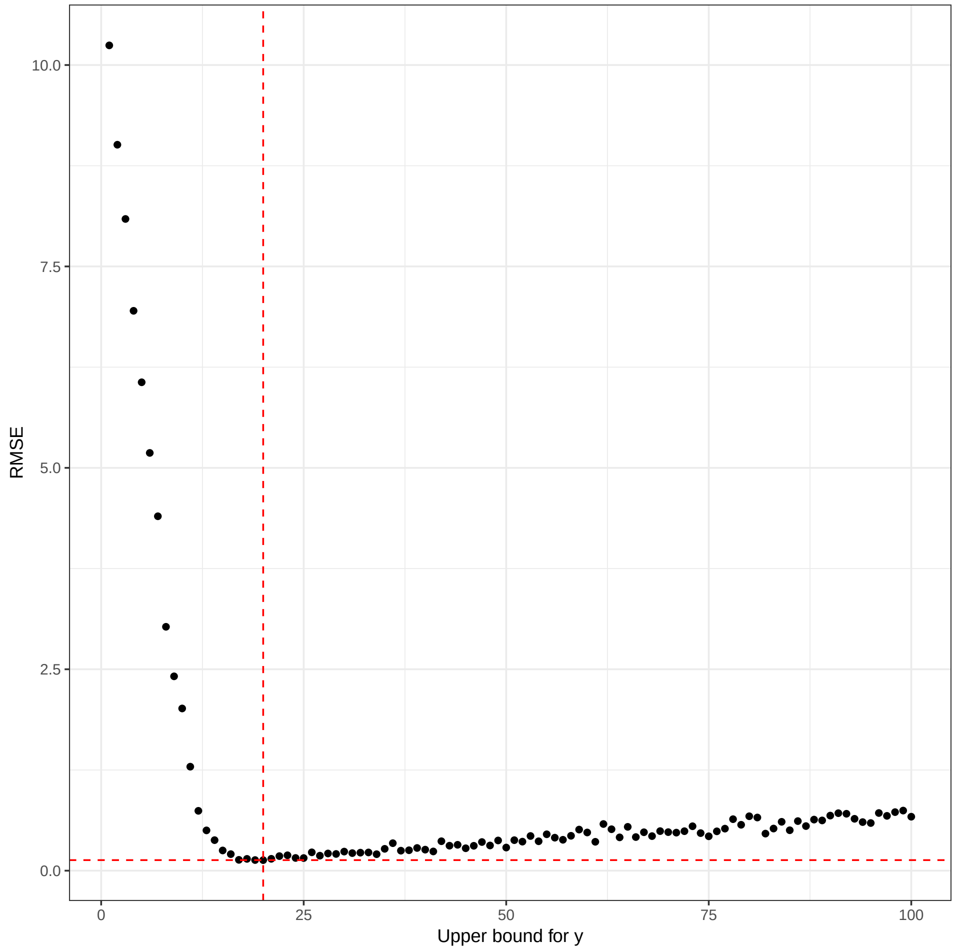
(e) -propose new mechanism -local sensitivity –> lower sensitivity in general, means thay we draw less noisy means -BOOM -or a diffeent statistic, different mechanism -see section notes

Question 3

(a) I break up the regression coefficient and intercept release mechanism into four differentially private subroutines. We showed in class that simply computing a differentially private version of the coefficient directly would be useless because the global sensitivty approaches ∞.

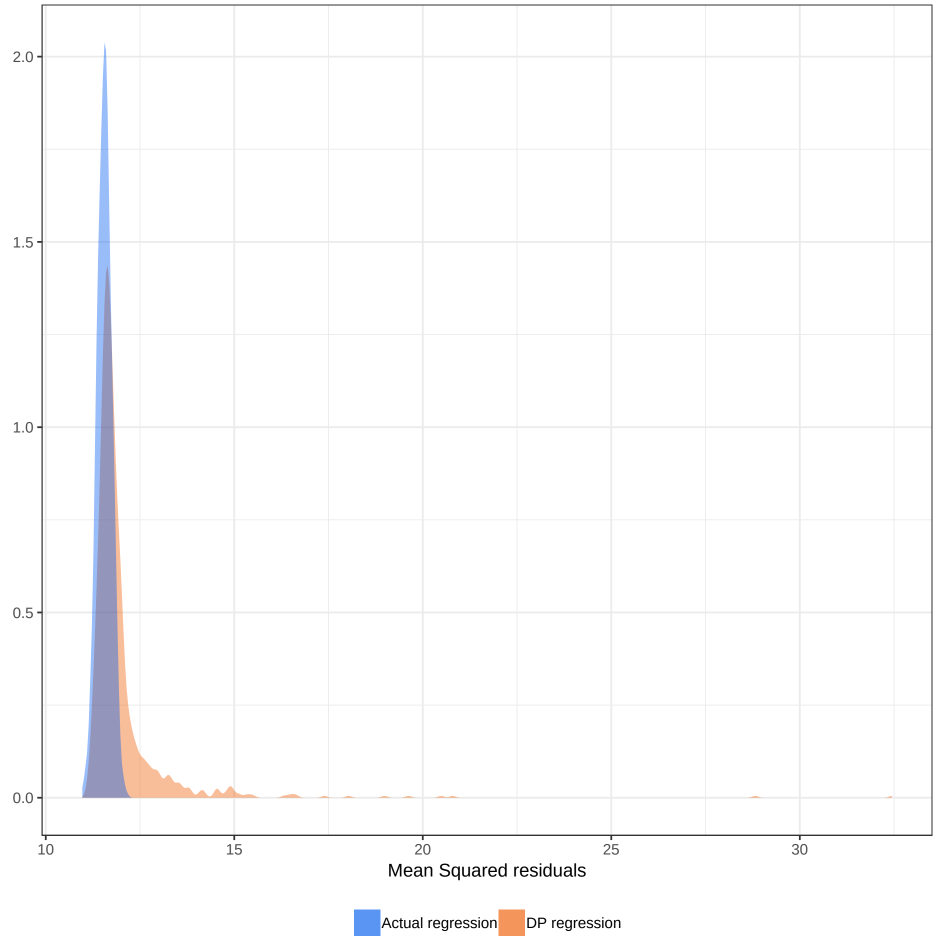
First, I know that I can write the coefficient of the regression as $\frac{S_{xy}}{S_{xx}}$. Thus, computing a differentially private version of the coefficient requires two subroutines: one for S_{xy} and another for S_{xx}. Next, to calculate the intercept, I can use the formula $\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$. Since our β is now differentially private, we can use the post-processing property to use it in this calculation with no extra privacy loss. However, we also need to compute a differentially private version of $\bar{x}$, and $\bar{y}$. To do this, I'll use the clamped Laplace mean release mechanism.

Furthermore, I will clamp both x and y as a part of this regression release mechanism. In Question 2 I found 19 to be a good upper bound value for x, and I'll continue to use that here. To get a good upper bound for y, I repeated the same process that I used for x but will also feed the data randomly drawn from a Poisson distribution into the noisy linear function and find the best upper bound for y that reduces the RMSE between the actual and noisy y values from the clamped Laplace mean release mechanism. As shown in the plot below, I find that 17 is the optimal upper bound for y.



(b) I generate 1,000 different datasets, perform my regression release on each dataset (splitting my budget evenly across the four subroutines), and then compare the performance of my noisy regression parameters to the performance of the true regression parameters by computing the mean-squared residuals for the original data. Here I consider the true regression to be the one computed on the clipped data so that my comparison focuses specifically on the effect of the added noise and not on the clipping procedure.

The plot below shows the distributions of the mean-squared residuals for the private and non-private regressions (the plot shows kernel density estimates). In general, the distribution of the mean-squared residuals for the private regression has much longer tail, implying that this approach has a larger expected error and could potentially lead to an enormous, utility-diminshing amount of error.



(c) Now I run a grid search to attempt to identify a better partitioning of the privacy budget that leads to lower error. To set up my grid, I consider weights for each statistic ranging from 0.1 to 0.9 by steps of 0.1. I then apply a softmax function to ensure that each partition of epsilon adds up to 1 - the total privacy budget for this problem. In total, I consider 6,561 partitions, although some are redundant.

Question 4

First we can write out

$$\frac{\Sigma_{i=1}^{n} P[A(M(X))_i = X_i]}{n}$$

$$\frac{\Sigma_{i=1}^{n} P[A(M(X_i, X_{-i})) = X_i]}{n}$$

By the definition of (ϵ, δ)-differential privacy, we have

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

- average of indicator functions?

Appendix

I put the code for all of my analyses here. You can also find it on Github.

Question 2

```
# c
rmse <- function(pred, true){ return(sqrt(mean((pred-true)^2))) }

n = 200
epsilon = 0.5
b_vals = seq(from=1, to=100, by=1)
n_sims <- 100

results <- matrix(NA, nrow=(length(b_vals)*n_sims), ncol=4)
i = 1
for (b in b_vals){
  dat <- poissonDGP(n)
  for (j in 1:n_sims){
    DPrelease <- laplaceClampMeanRelease(dat, epsilon, a=0, b=b)
    results[i,1] <- b
    results[i,2] <- j
    results[i,3] <- DPrelease$release
    results[i,4] <- DPrelease$true
    i = i + 1
  }
}
results_df <- data.frame(results)
names(results_df) <- c("b", "sim", "release", "true")
avg_results_df <- results_df %>% group_by(b) %>% summarise(rmse = rmse(release, true))

q2_plot <- ggplot(data=avg_results_df, aes(x=b, y=rmse)) +
  geom_point() + geom_hline(yintercept = min(avg_results_df$rmse), col="red", lty=2) +
  geom_vline(xintercept = avg_results_df[which.min(avg_results_df$rmse), ]$b, col="red", lty=2) +
  labs(x="Upper bound", y="RMSE") + theme_bw()
```

Question 3

```
# a
poissonDGP <- function(n){ return(rpois(n, lambda=10)) }
noisyLinearDGP <- function(x, n, alpha, beta, mu=0, sd=1) { return(beta*x + alpha + rnorm(n, mu, sd)) }

sgn <- function(x) {     # function borrowed from class
  return(ifelse(x < 0, -1, 1))
}

rlap = function(mu=0, b=1, size=1) {     # function borrowed from class
  p <- runif(size) - 0.5
  draws <- mu - b * sgn(p) * log(1 - 2 * abs(p))
  return(draws)
}
```

```
clip <- function(x, lower, upper){     # function borrowed from class
  x.clipped <- x
  x.clipped[x.clipped<lower] <- lower
  x.clipped[x.clipped>upper] <- upper
  return(x.clipped)
}

rmse <- function(pred, true){ return(sqrt(mean((pred-true)^2))) }

laplaceClampMeanRelease <- function(x, epsilon, a=0, b=1){     # from q2
  n <- length(x)
  sensitivity <- (a - b)/n
  scale <- sensitivity / epsilon

  x.clipped <- clip(x, a, b)
  clipped.mean <- mean(x.clipped)
  noisy.mean <- clipped.mean + rlap(mu=0, b=scale, size=1)
  release.mean <- clip(noisy.mean, a, b)
  true.mean <- mean(x)

  return(list(release=release.mean, true=true.mean))
}

regressionRelease <- function(y, x, ylower=0, yupper=17, xlower=0, xupper=19, eplsilon, partition){
  x <- clip(x, xlower, xupper)
  y <- clip(y, ylower, yupper)

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

  scale.Sxy <- sens.Sxy / (epsilon*partition$Sxy)
  scale.Sxx <- sens.Sxx / (epsilon*partition$Sxx)

  true.beta <- sum((x - mean(x))*(y - mean(y))) / sum((x - mean(x))^2)
  true.alpha <- mean(y) - true.beta*mean(x)

  release.Sxy <- sum((x - mean(x))*(y - mean(y)))  + rlap(mu=0, b=scale.Sxy, size=1)
  release.Sxx <- sum((x - mean(x))^2) + rlap(mu=0, b=scale.Sxx, size=1)
  release.beta <- release.Sxy/release.Sxx

  release.x.bar <- laplaceClampMeanRelease(x, epsilon*partition$x.bar, a=xlower, b=xupper)$release
  release.y.bar <- laplaceClampMeanRelease(y, epsilon*partition$y.bar, a=ylower, b=yupper)$release
  release.alpha <- release.y.bar - release.beta*release.x.bar

  release.mean.sq.residuals <- mean((y - release.beta*x - release.alpha)^2)
  true.mean.sq.residuals <- mean((y - true.beta*x - true.alpha)^2)

  return(list(release.beta=release.beta,
              release.alpha=release.alpha,
              true.beta=true.beta,
              true.alpha=true.alpha,
              release.mean.sq.residuals=release.mean.sq.residuals,
              true.mean.sq.residuals=true.mean.sq.residuals))
```

```
}

# get optimal upper bound for y
n = 200
epsilon = 1
b_vals = seq(from=1, to=100, by=1)
n_sims <- 100

results_y <- matrix(NA, nrow=(length(b_vals)*n_sims), ncol=4)
i = 1
for (b in b_vals){
  dat <- poissonDGP(n)
  y <- noisyLinearDGP(dat, n, alpha=1, beta=1, mu=0, sd=1)
  for (j in 1:n_sims){
    DPrelease <- laplaceClampMeanRelease(y, epsilon, a=0, b=b)
    results_y[i,1] <- b
    results_y[i,2] <- j
    results_y[i,3] <- DPrelease$release
    results_y[i,4] <- DPrelease$true
    i = i + 1
  }
}
results_y_df <- data.frame(results_y)
names(results_y_df) <- c("b", "sim", "release", "true")
avg_results_y_df <- results_y_df %>% group_by(b) %>% summarise(rmse = rmse(release, true))

q3_plot1 <- ggplot(data=avg_results_y_df, aes(x=b, y=rmse)) +
  geom_point() + geom_hline(yintercept = min(avg_results_y_df$rmse), col="red", lty=2) +
  geom_vline(xintercept = avg_results_y_df[which.min(avg_results_y_df$rmse), ]$b, col="red", lty=2) +
  labs(x="Upper bound for y", y="RMSE") + theme_bw()
pdf("plots/q3_plot1.pdf", width=8, height=8)
q3_plot1
dev.off()

# b
equal_partition <- list(Sxy=0.25, Sxx=0.25, x.bar=0.25, y.bar=0.25)
n = 1000
alpha = beta = eplison = sd = 1
n_sims = 1000

results_reg <- matrix(NA, nrow=n_sims, ncol=6)
for (i in 1:n_sims){
  x <- poissonDGP(n)
  y <- noisyLinearDGP(dat, n, alpha=1, beta=1, mu=0, sd=1)
  DPrelease <- regressionRelease(y, x, ylower=0, yupper=17, xlower=0, xupper=19, eplsilon, equal_partiti
  results_reg[i,1] <- DPrelease$release.beta
  results_reg[i,2] <- DPrelease$release.alpha
  results_reg[i,3] <- DPrelease$true.beta
  results_reg[i,4] <- DPrelease$true.alpha
  results_reg[i,5] <- DPrelease$release.mean.sq.residuals
  results_reg[i,6] <- DPrelease$true.mean.sq.residuals
}
results_reg_df <- data.frame(results_reg)
names(results_reg_df) <- c("release.beta", "release.alpha", "true.beta",
```

```
                           "true.alpha", "release.mean.sq.residuals", "true.mean.sq.residuals")

semi.blue <- rgb(0,90,239,50,maxColorValue=255)
semi.red  <- rgb(239,90,0,200,maxColorValue=255)
q3_plot2 <- ggplot(data=results_reg_df) +
  geom_density(aes(x=release.mean.sq.residuals, fill="DP regression"), alpha=0.4, colour=NA) +
  geom_density(aes(x=true.mean.sq.residuals, fill="Actual regression"), alpha=0.4, colour=NA) +
  labs(x="Mean Squared residuals", y="") + theme_bw() +
  scale_fill_manual(values=c(semi.blue, semi.red)) +
  theme(legend.position="bottom", legend.title=element_blank())

# d
(more code to come)
```