HW 1: Reidentification, Reconstruction and Membership Attacks

CS 208 Applied Privacy for Data Science, Spring 2019
Lipika Ramaswamy

February 26, 2019

Note: code for all the problems can be found in the Appendix. R files can be found on Github (hw1q2.R and hw1q3.R) [https://github.com/lipikaramaswamy/cs208_lr/tree/master/homework].

1. Reidentification Attack

The PUMS dataset of 25,766 rows contains data for individuals in Georgia (state 13) within the fips code of 13121. There are seven unique public use microdata areas (PUMAs) in this dataset. The other columns in the dataset include age (ranging from 18 to 93), education on a fifteen point scale, income in dollars rounded to the nearest 10, and binary variables for sex, latino, black, asian, married, divorced, uscitizen, children, disability, military service, employed and english ability.

To construct a record linkage reidentification attack, one approach would be to determine the marginal probability of an individual in the data having each attribute described above, as shown in Table 1. The probability of an individual, A, living in a given PUMA, say 1101 for this example, is also known. Consider an attacker wants to identify A, a disabled divorced Asian female who lives in PUMA 1101 and has good english ability in this dataset. Assuming that the probability of belonging to each group is independent, a simple calculation shows that A is the only person in this dataset of 25,766 with those attributes, and she can be easily identified.

$$\text{Number of individuals with A's attributes in sample} = (0.53 \times 0.03 \times 0.11 \times 0.22 \times 0.96 \times 0.12) \times 25,766 = 1.14 \approx 1$$

Given that the sample represents 5% of the population of Georgia, we can apply these marginal probabilities to estimate the number of individuals with A's attributes in the population, as below. This shows that there are roughly 23 people in the population with these attributes.

Table 1: Marginal probabilities of binary columns

	x
sex	0.53
asian	0.03
divorced	0.11
disability	0.22
englishability	0.96
puma _1101	0.12

$$\text{Number of individuals with A's attributes in population} = (0.53\times0.03\times0.11\times0.22\times0.96\times0.12)\times\left(\frac{25,766}{0.05}\right)$$

$$= 22.8 \approx 23$$

For individuals who might not be as easily identifiable with just a handful of marginal probabilities of attributes and for attributes that may not truly be independent, the joint probabilities of different attributes can be considered to perform a reidentification attack.

2. Reconstruction Attack

In this reconstruction attack, an adversary is attempting to learn sensitive information about individuals in the dataset from which aggregates of specified groups are accessible. The sensitive information is whether an individual in the sample of 100 individuals from Fulton County is a US citizen. Based on results seen in class, it is known that when the error added to the answer is less than $\sqrt{n}$, the fraction of recovered bits is $1 - o(1)$. The amount of error added was varied, and the impact can be seen in the discussion that follows.

Below is a description of reconstruction attacks in the face of each defense described above.

1. Rounding the sum to the nearest multiple of R for R$\in\{1,2,\ldots,100\}$:

   The results of rounding are shown in Figure 1.

   As can be seen from the plot of root mean square error (RMSE) for the range of R values, the RMSE increases from zero, sees a slight decline right around R=25, and climbs up, and then decreases around R=50, before continuing to be quite positive. The decrease in RMSE around R=50 is not surprising, given that the sum of US citizen associated with each random subset of the 100 rows were mostly in the range of 40 to 60.

   The accuracy of the reconstruction attack was perfect for R=1. Thereafter, there was a decline in accuracy until roughly R=35, then an increase until roughly R=62, followed by a decrease. This trend is in keeping with the trend for RMSE.

   The plot of accuracy vs. RMSE shows that for reconstruction attacks with a higher accuracy, the RMSE is lower, demonstrating the privacy-utility tradeoff. This makes sense, as the lower the RMSE, i.e. the difference between the true sum and the noisy sum, the higher the ability of the adversary to reconstruct the sensitive bits of individuals in the sample. As RMSE increases, the accuracy of the reconstruction attack decreases, as seen in the plots.

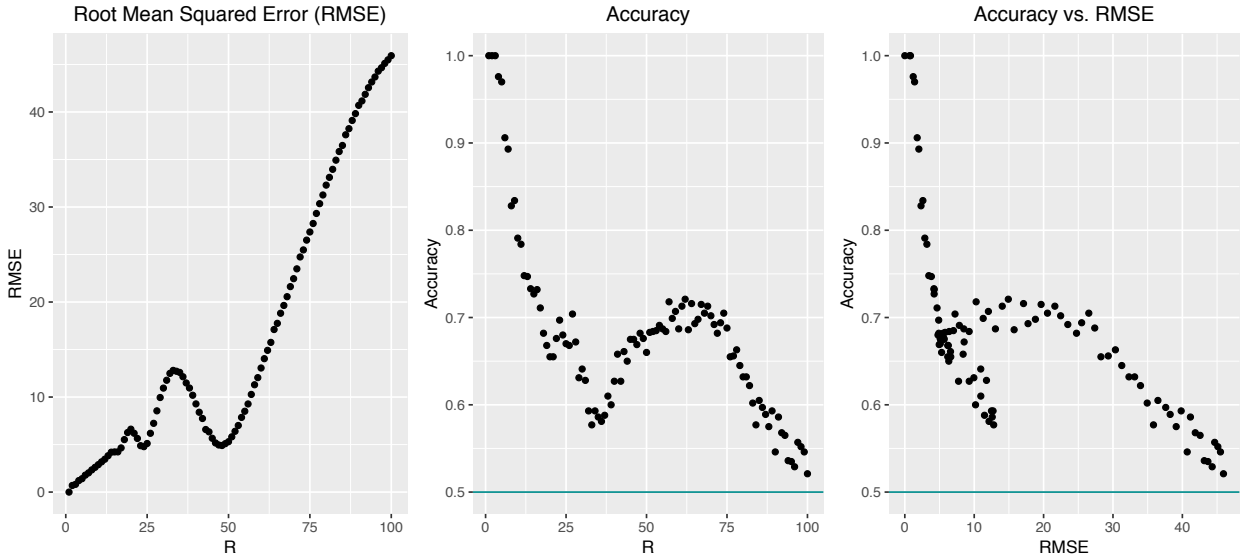
2. Adding Gaussian noise with zero mean and standard deviation $\sigma \in\{1,2,\ldots,100\}$ to the sum:

   The results of Gaussian noise are shown in Figure 2.

   Adding Gaussian noise to the sum causes the RMSE to increase linearly with the standard deviation, σ, of the noise added. This makes sense as the sums become large and very different from their true values.

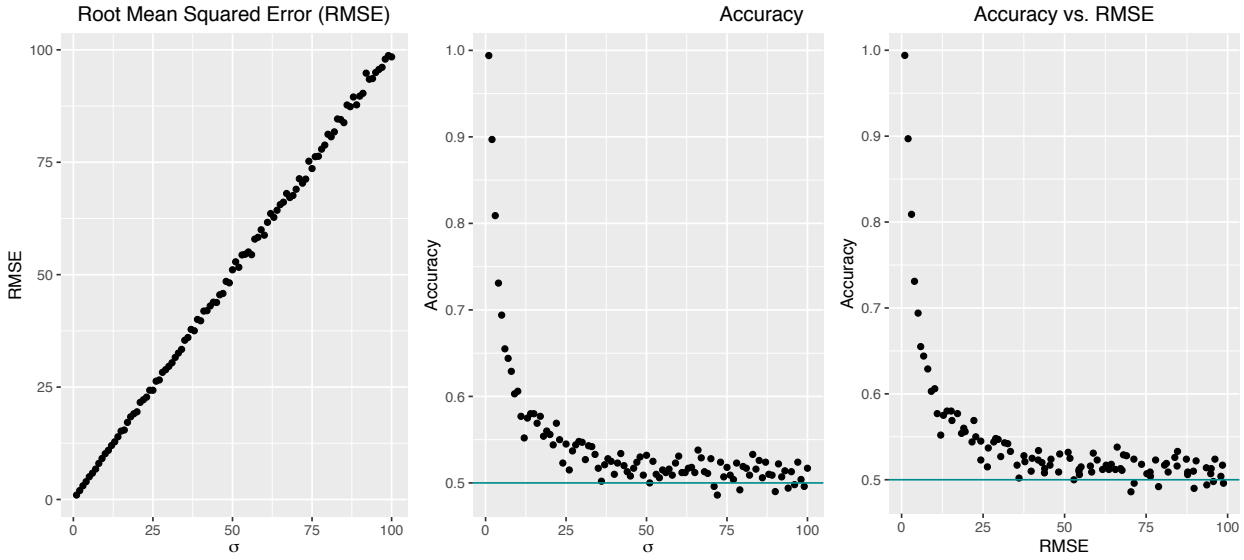
   The accuracy of the reconstruction attack started out near perfect for $sigma = 1$, and then saw an exponential decay as the standard deviation of the added noise increases. This is quite intuitive as well, given that the sums start to become far from their true value rather quickly.

Figure 1: RMSE and accuracy plots for sums returned with rounding defense



The privacy utility trade off is well demonstrated in this case as well. As the accuracy of the responses decrease, i.e. the privacy improves, the RMSE increases, i.e. the statistics released are not as useful.

Figure 2: RMSE and accuracy plots for sums returned with rounding defense



3. Subsampling a set T consisting of t out of the $n = 100$ rows, with $t \in \{1,2,\ldots,100\}$ and scaling the sum:

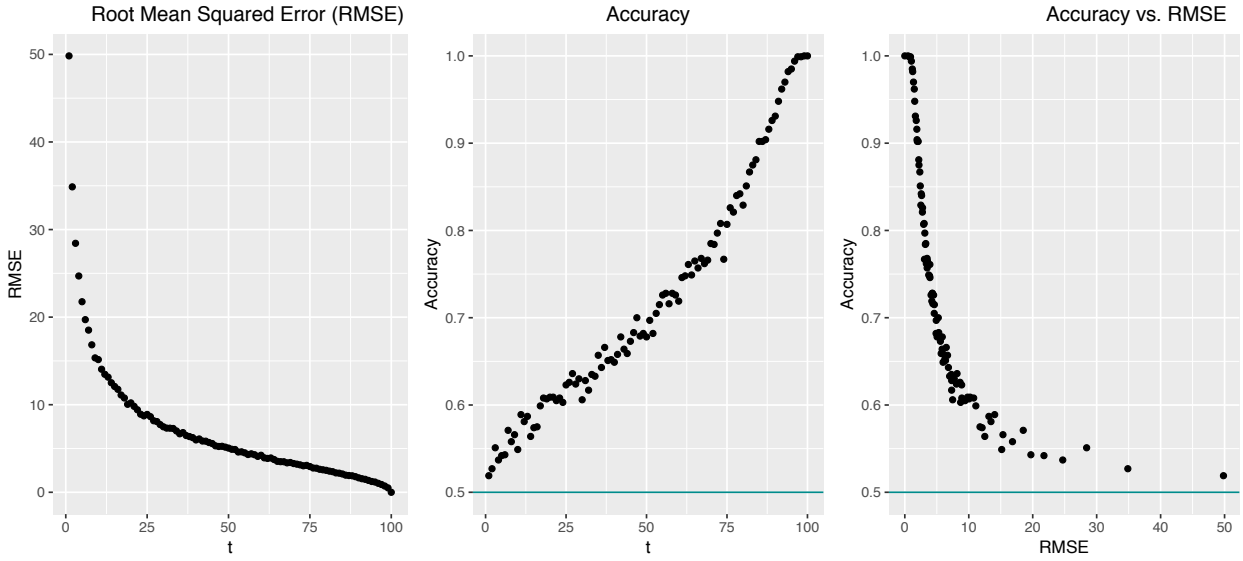
   The results of subsampling are shown in Figure 3.

   The RMSE of the reconstruction attack decays somewhat exponentially as the size of the subsample (t) increases. This makes sense as increasing the size of the subsample reduces the distance between the noisy sum and true sum of the sensitive attribute.

   The accuracy of the reconstruction attack increases as t increases. This makes sense for the same reason as above.

   The privacy-utility trade off is also well demonstrated in this case. As the accuracy of the reconstruction attack decreases, the RMSE increases.

Figure 3: RMSE and accuracy plots for sums returned with subsampling defense



3. Membership Attack

Even when reconstruction attacks start to fail (low accuracies around 0.5), membership attacks are still possible. For a membership attack, when $m = n^2$ queries (or attributes), the system can be vulnerable to reconstruction attacks. Here, we found values for the parameters of the defenses explored above where the reconstruction starts to fail. For rounding, R=100 was selected as at this point, the accuracy of the reconstructed sensitive attribute was the lowest (as seen in Figure 1). For Gaussian noise, values of σ around 50 start to result in failure to reconstruct. For subsampling, subsets of size less than 5 yield the lowest accuracy. These values were selected going forward to the membership attack, along with values that yielded higher accuracy of reconstruction in the reconstruction attack.

Below is a summary of the trends noticed when increasing the number of attributes available for the membership attack under the three different defenses:

1. Rounding Defense:

   The true positive rate for the membership attack reaches 100% for R=10 when the number of attributes is around 3000. This is the case when there isn't enough noise for the reconstruction attack to fail completely. Comparing this to the true positive rate for the membership attack with R=100, we find that this reaches 100% around 3,750. This makes sense, as increasing the noise in the returned answer (a) requires more attributes of the individual to be known to reach 100% true positive probability. Thus rounding doesn't work too well as a defense mechanism even for high values of R. These results are visualized in Figure 4.

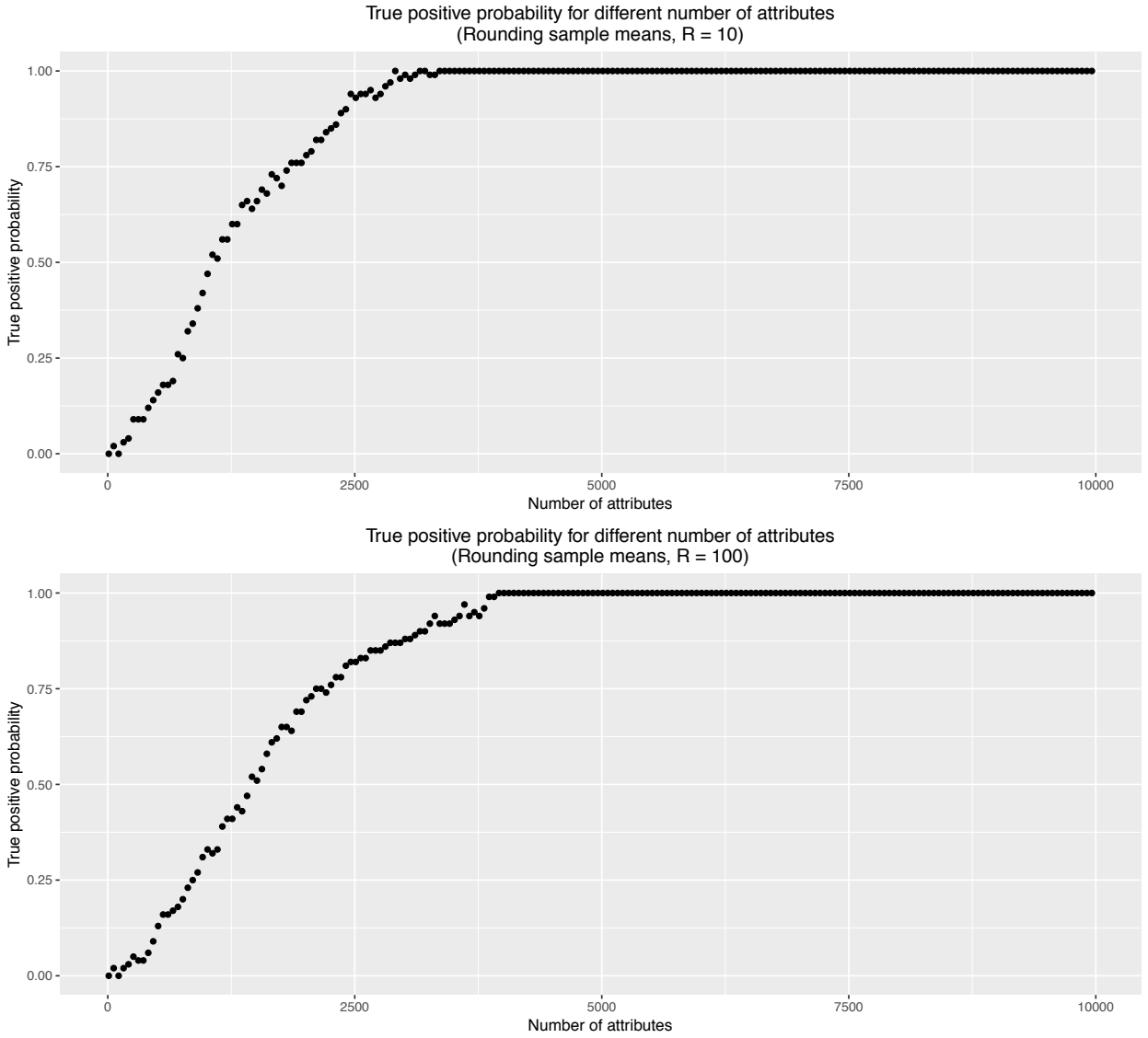
2. Gaussian Noise:

   The true positive rate for the membership attack reaches about 80% for $\sigma = 10$ when gaussian noise is included in the returned answer and the number of attributes is 10,000. When $\sigma = 45$, which was identified as the point where reconstruction fails, the true positive probability of the membership attack reaches 100% when the number of attributes increases to 3,750. This is interesting, as I would expect the true positive rate to be reached for a higher number of attributes for higher σ. I'm not sure what's going on here, but given the observation for smaller sigma, I infer that since the true positive probability for the membership attack never reaches 100% even up to 10,000 attributes, which is better than the case for rounding. These results are visualized in Figure 5.

3. Subsampling:

   For $t = 2$, the true positive rate for this membership attack stays at 2% as attributes are added until we reach number of attributes greater than 9000, when it jumps to 3%. This makes sense as the answers returned by this mechanism are so noisy when we are only subsampling two rows of data, that it's hard to ever determine whether Alice is in the sample or not based on our test statistic. When the size of the subsample is increased to 50, the plot looks a lot more similar to what was viewed in the other two cases, where the true positive probability reaches 0.5 when the number of attributes increases past 2500. Thus subsampling is proving to be the best defense mechanism for small values of t, though it is worth noting that the utility of the answers is very poor given the small subsample size. These results are visualized in Figure 6.

Figure 4: True positive rate for membership attack on means with rounding defense



4. Project ideas:

   I am particularly interested in working on privacy in the context of machine learning algorithms. Specifically, I found a paper by Abadi et al. [https://arxiv.org/pdf/1607.00133.pdf] that discuss neural nets trained on stochastic gradient descent, and privacy-preserving versions of the algorithm that model the privacy loss as a random variable. It would be interesting to implement this. I was also intrigued by our discussion of Reza Shokri's slides on memberbership inference attacks, and Ariel's comment that her lab had not been able to replicate their results. I looked up more of his work and found a recent paper on white-box inference attacks[https://www.comp.nus.edu.sg/~reza/files/Shokri-SP2019.pdf]. I think it could be really interesting to attemp to replicate their methodology and if not, try to understand what makes it difficult. Looking forward to your feedback on these ideas!

Figure 5: True positive rate for membership attack on means with added Gaussian noise

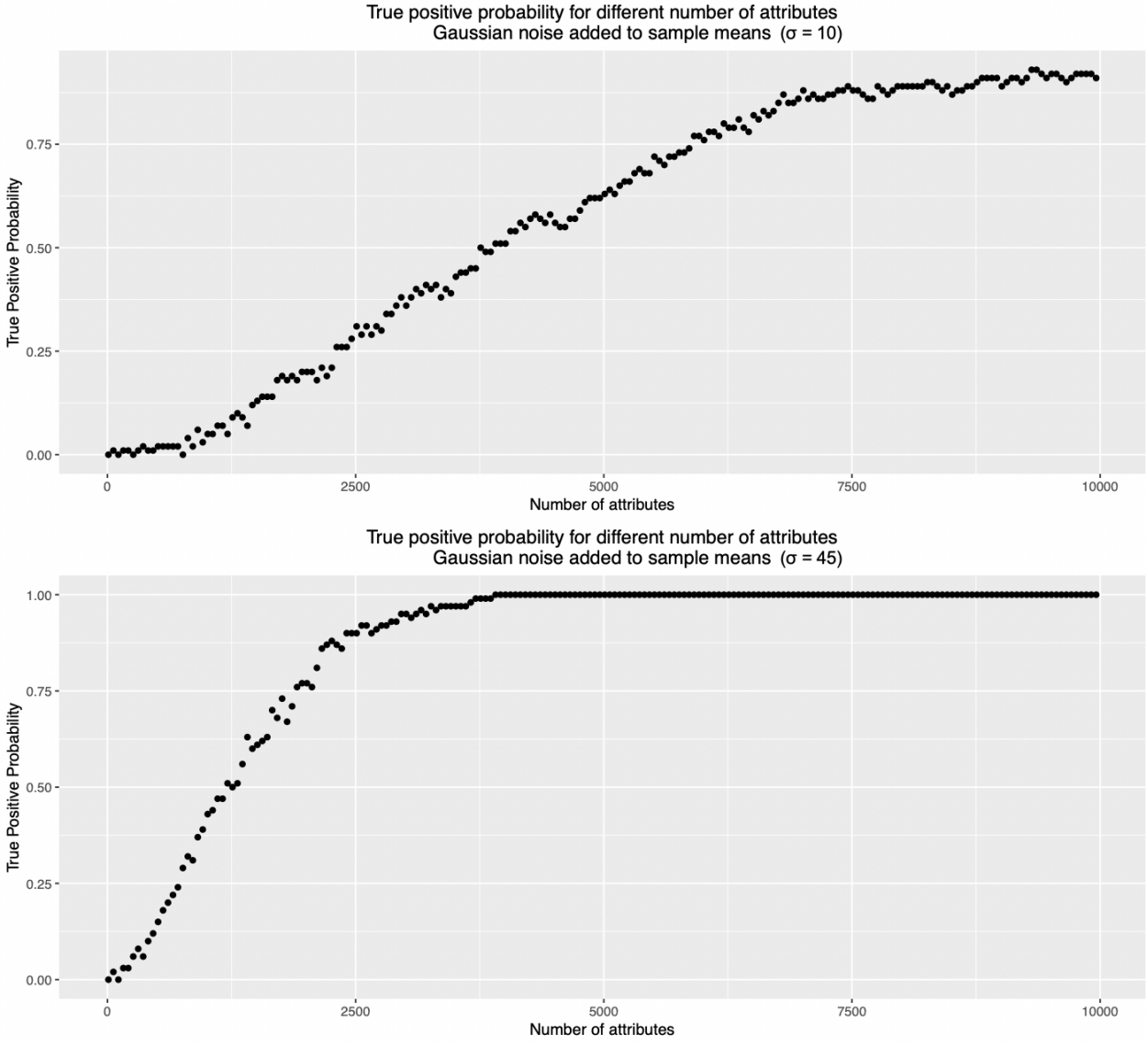
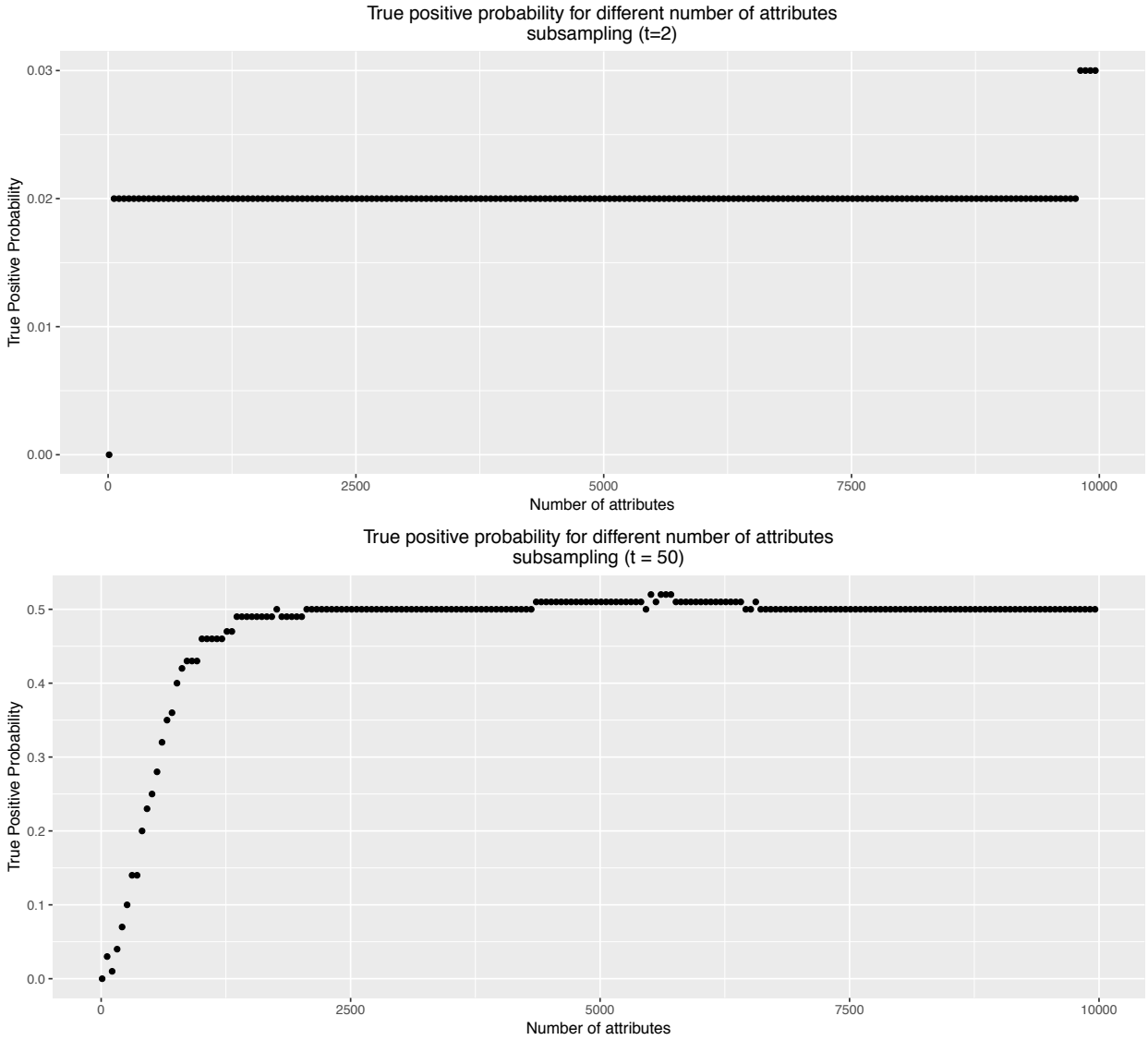


Figure 6: True positive rate for membership attack on means with added Gaussian noise



HW 1 Appendix

Reconstruction Attack

```
## import libraries
library(caret)
library(plyr)
library(dplyr)
library(ggplot2)
library(gridExtra)

# ------------------------------------------------ #

#### Parameters ####

## number of queries
k.trials <- 200
## number of rows in dataset
n <- 100

# ------------------------------------------------ #



### ALL THINGS DATA - Read in, subset, set aside sensitive data ###
setwd("/Users/lipikaramaswamy/Documents/Harvard/CS208/cs208_lr/")


## read in population

population <- read.csv('data/FultonPUMS5full.csv')
mean.uscitizen.true = mean(population$uscitizen)


## read in sample
pums <- read.csv("data/FultonPUMS5sample100.csv")

# subset data to that which is available to attacker
available.pums = select(pums, uscitizen, sex, age, educ, latino, black, asian, married,
                        divorced,children,disability,militaryservice,employed,englishability)

# make new column to contain randomly hashed values that determine membership to the random sub
available.pums$subset.indicator<-NA

# set aside sensitive data:
```

```
sensitive.data <- pums[, "uscitizen"]

# ------------------------------------------------ #

### Setup to build random subsets of data ###

# Chose random large prime number
P = 491

# Make vector of all integers up to P
prime.options <- seq(from=0, to=P-1, by=1)

# ------------------------------------------------ #

### BUILD QUERY ###

runQuery <- function(df, rounding = FALSE, R = 0, gaussian = FALSE, gaussian.sigma = 0, subsam

  ## random subset creation
  r = sample(prime.options, size=13, replace = FALSE)
  mat.of.obs = as.matrix(available.pums[,2:14])
  p = (mat.of.obs %*% r) %% P %% 2
  df$subset.indicator = p

  ## subsetting and returning the sum
  subset = df[df$subset.indicator == 1,]
  sum <- sum(subset$uscitizen)
  index = as.numeric(rownames(subset))

  ## round if specified
  if (rounding == TRUE){
    round <- round_any(sum, R)
    return(list(sum=round, index=index, truesum = sum))
  }

  ## add gaussian noise if specified
  if (gaussian == TRUE){

    noisy <- sum + rnorm(1,0,(gaussian.sigma))
    return(list(sum=noisy, index=index, truesum = sum))
  }

  ## subsample and scale query result if specified
  if (subsample == TRUE){

    subsample.index <- sample(x=1:nrow(df), size=subsample.size, replace = FALSE)
    subset <- df[subsample.index,]
    subsetsum <- sum(subset$subset.indicator * subset$uscitizen) * (nrow(df)/subsample.size)
```

```
    return(list(sum=subsetsum, index=index, truesum = sum))
  }

  ## if none of the defense mechanisms are specified, return true sum
  if (rounding == FALSE & rounding == FALSE & gaussian == FALSE){
    return(list(index = index, truesum = sum))
  }
}

# ------------------------------------------------ #

### RUNNING ATTACKS - MULTIPLE PARAMETERS, MULTIPLE EXPERIMENTS ###

## set varnames
xnames <- paste("x", 1:n, sep="")
varnames<- c("y", xnames)

## Make formula
formula <- paste(xnames, collapse=" + ")
formula <- paste("y ~ ", formula, "-1")
formula <- as.formula(formula)

## matrix to contain results of the queries and indices
history.rounding <- matrix(NA, nrow=k.trials, ncol=100+2)
history.gaussian <- matrix(NA, nrow=k.trials, ncol=100+2)
history.subsamp <- matrix(NA, nrow=k.trials, ncol=100+2)

## set parameter ranges to loop thru
parameter.range <- seq(from=1, to=100, by=1)
RMSE.matrix <- matrix (, nrow = 100, ncol = 10)
acc.matrix <- matrix (, nrow = 100, ncol = 10)

### Run attack for ROUNDING
for(b in 1:10){

  for(a in parameter.range){
    ## build matrix for regression

    for(i in 1:k.trials){
      res <- runQuery(df=available.pums, rounding = TRUE, R = a)
      indicator <- 1:n %in% res$index
      indicator <- as.numeric(indicator)
      history.rounding[i,] <- c(res$truesum, res$sum, indicator)
    }

    ## Convert matrix into data frame
    release.data.rounding <- as.data.frame(history.rounding[,2:102])
    names(release.data.rounding) <- varnames
```

```
    ## Run reg and get estimates
    output.rounding <- lm(formula, data=release.data.rounding)
    estimates.rounding <- output.rounding$coef

    RMSE.matrix[a,b] <- postResample(history.rounding[,1], history.rounding[,2])[1]
    correct.preds <- ((estimates.rounding>0.5) == sensitive.data)
    acc.matrix[a,b] <- sum(correct.preds)/100

  }

  RMSE.rounding <-rowMeans(RMSE.matrix)
  acc.rounding <-rowMeans(acc.matrix)

}

### Run attack for GAUSSIAN NOISE

for(b in 1:10){
  for(a in parameter.range){
    ## build matrix for regression
    for(i in 1:k.trials){
      res <- runQuery(df=available.pums, gaussian = TRUE, gaussian.sigma = a)
      indicator <- 1:n %in% res$index
      indicator <- as.numeric(indicator)
      history.gaussian[i,] <- c(res$truesum, res$sum, indicator)
    }

    ## Convert matrix into data frame
    release.data.gaussian <- as.data.frame(history.gaussian[,2:102])
    names(release.data.gaussian) <- varnames

    ## Run reg and get estimates
    output.gaussian <- lm(formula, data=release.data.gaussian)
    estimates.gaussian <- output.gaussian$coef

    RMSE.matrix[a,b] <- postResample(history.gaussian[,1], history.gaussian[,2])[1]
    correct.preds <- (estimates.gaussian>0.5) & (sensitive.data==1) | (estimates.gaussian<0.5)
    acc.matrix[a,b] <- sum(correct.preds)/100}

  RMSE.gaussian <-rowMeans(RMSE.matrix)
  acc.gaussian <-rowMeans(acc.matrix)
}

### Run attack for SUBSAMPLING

for(b in 1:10){
```

```
  for(a in parameter.range){
    ## build matrix for regression

    for(i in 1:k.trials){
      res <- runQuery(df=available.pums, subsample = TRUE, subsample.size = a)
      indicator <- 1:n %in% res$index                          # convert indices into a series o
      indicator <- as.numeric(indicator)
      history.subsamp[i,] <- c(res$truesum, res$sum, indicator)                    # save into
    }

    ## Convert matrix into data frame
    release.data.subsamp <- as.data.frame(history.subsamp[,2:102])
    names(release.data.subsamp) <- varnames

    ## Run reg and get estimates
    output.subsamp <- lm(formula, data=release.data.subsamp)
    estimates.subsamp <- output.subsamp$coef

    RMSE.matrix[a,b] <- postResample(history.subsamp[,1], history.subsamp[,2])[1]
    correct.preds <- (estimates.subsamp>0.5) & (sensitive.data==1) | (estimates.subsamp<0.5) &
    acc.matrix[a,b] <- sum(correct.preds)/100
  }

  RMSE.subsamp <-rowMeans(RMSE.matrix)
  acc.subsamp <-rowMeans(acc.matrix)

}


## PLOT RESULTS

rounding.for.plots = data.frame(parameter.range, RMSE.rounding, acc.rounding)

p1 <- ggplot(rounding.for.plots, aes(x = parameter.range, y = RMSE.rounding)) +
  geom_point() +
  labs(x = "R", y = 'RMSE', title = 'Root Mean Squared Error (RMSE)') +
  theme(plot.title = element_text(hjust = 0.5))


p2 <- ggplot(rounding.for.plots, aes(x = parameter.range, y = acc.rounding)) +
  geom_point() +
  geom_hline(yintercept = 0.5, color = "darkcyan") +
  labs(x = "R", y = 'Accuracy', title = 'Accuracy') +
  theme(plot.title = element_text(hjust = 0.5))

p3 <- ggplot(rounding.for.plots, aes(x = RMSE.rounding, y = acc.rounding)) +
  geom_point() +
  geom_hline(yintercept = 0.5, color = "darkcyan") +
```

```
  labs(x = "RMSE", y = 'Accuracy', title = 'Accuracy vs. RMSE') +
  theme(plot.title = element_text(hjust = 0.5))

plots.rounding = grid.arrange(p1, p2, p3, nrow = 1)

ggsave(filename = 'rounding_plots.pdf', plot = plots.rounding, width = 11, height = 5, units =


gaussian.for.plots = data.frame(parameter.range, RMSE.gaussian, acc.gaussian)

p4 <- ggplot(gaussian.for.plots, aes(x = parameter.range, y = RMSE.gaussian)) +
  geom_point() +
  labs(x = expression(paste(sigma)), y = 'RMSE', title = 'Root Mean Squared Error (RMSE)') +
  theme(plot.title = element_text(hjust = 0.5))

p5 <- ggplot(gaussian.for.plots, aes(x = parameter.range, y = acc.gaussian)) +
  geom_point() +
  geom_hline(yintercept = 0.5, color = "darkcyan") +
  labs(x = expression(paste(sigma)), y = 'Accuracy', title = 'Accuracy') +
  theme(plot.title = element_text(hjust = 0.5))

p6 <- ggplot(gaussian.for.plots, aes(x = RMSE.gaussian, y = acc.gaussian)) +
  geom_point() +
  geom_hline(yintercept = 0.5, color = "darkcyan") +
  labs(x = "RMSE", y = 'Accuracy', title = 'Accuracy vs. RMSE') +
  theme(plot.title = element_text(hjust = 0.5))

plots.gaussian = grid.arrange(p4, p5, p6, nrow = 1)

ggsave(filename = 'gaussian_plots.pdf', plot = plots.gaussian, width = 11, height = 5, units =

subsamp.for.plots = data.frame(parameter.range, RMSE.subsamp, acc.subsamp)

p7 <- ggplot(subsamp.for.plots, aes(x = parameter.range, y = RMSE.subsamp)) +
  geom_point() +
  labs(x = "t", y = 'RMSE', title = 'Root Mean Squared Error (RMSE)') +
  theme(plot.title = element_text(hjust = 0.5))

p8 <- ggplot(subsamp.for.plots, aes(x = parameter.range, y = acc.subsamp)) +
  geom_point() +
  geom_hline(yintercept =  0.5, color = "darkcyan") +
  labs(x = "t", y = 'Accuracy', title = 'Accuracy') +
  theme(plot.title = element_text(hjust = 0.5))

p9 <- ggplot(subsamp.for.plots, aes(x = RMSE.subsamp, y = acc.subsamp)) +
  geom_point() +
  geom_hline(yintercept = 0.5, color = "darkcyan") +
  labs(x = "RMSE", y = 'Accuracy', title = 'Accuracy vs. RMSE') +
```

```
  theme(plot.title = element_text(hjust = 0.5))

plots.subsampling = grid.arrange(p7, p8, p9, nrow = 1)

ggsave(filename = 'subsampling_plots.pdf', plot = plots.subsampling, width = 11, height = 5, u
```

Membership Attack

```
## CS208
## Q3 - Membership attack on PUMS

rm(list=ls())

## Import packages
library(caret)
library(plyr)
library(dplyr)
library(ggplot2)
library(gridExtra)

### GET DATA AND PREPARE IT
setwd("/Users/lipikaramaswamy/Documents/Harvard/CS208/cs208_lr/")

## read in full datasets
sample.full <- read.csv("data/FultonPUMS5sample100.csv")

## select only PUB cols
sample = select(sample.full, sex, latino, black, asian, married, age, educ,
                divorced,children,disability,militaryservice,employed,englishability)

## hash random predicates
P <- 105701
list.of.rs <- replicate(10000, sample(0:(P-1),size=13), simplify=TRUE)

## make 10000 attributes in sample
mat.of.obs.samp = as.matrix(sample)
new.samp = (mat.of.obs.samp %*% list.of.rs) %% P %% 2

# population means would be 0 given how the hash works
population.mean = rep(0, 10000)

### FUNCTIONS

## query the dataset and return sample means with the three defense mechanisms (rounding, gauss
membershipQuery <- function(samp,
                            rounding = FALSE, R,
                            gaussian = FALSE, gaussian.sigma,
```

```
                              subsample = FALSE, subsample.size){
  sample.sum.true = as.vector(colSums(samp, na.rm = FALSE, dims = 1))

  ## round if specified
  if (rounding == TRUE){
    round <- round_any(sample.sum.true, R)
    means <- round /100
    means <- 2*(means-0.5)
    return(means)
  }

  ## add gaussian noise if specified
  if (gaussian == TRUE){
    noisy <- sample.sum.true + rnorm(1,0,(gaussian.sigma))
    means <- noisy / 100
    means <- 2 * (means - 0.5)
    return(means)
  }

  ## subsample and scale query result if specified
  if (subsample == TRUE){
    subsample.index <- sample(x=1:nrow(samp), size=subsample.size, replace = FALSE)
    subset <- samp[subsample.index,]
    means <- as.vector(colMeans(subset, na.rm = FALSE, dims = 1))
    means <- 2*(means-0.5)
    return(means)
  }
}

# Dwork et al. test statistic using population means
test.Dwork <- function(alice, sample.mean, population.mean){
  test.statistic <- sum(alice * sample.mean) - sum(population.mean * sample.mean)
  return(test.statistic)
}

## A utility function to create data from the population
rmvbernoulli <- function(n=1, prob){
  history <- matrix(NA, nrow=n, ncol=length(prob))
  for(i in 1:n){
    x<- rbinom(n=length(prob), size=1, prob=prob)
    x[x==0] <- -1                                  # Transform from {0,1} to {-1,1}
    history[i,] <- x
  }
  return(history)
}

#### PARAMETERS
n.attributes = ncol(sample.augmented)
```

```
my.alpha <- 0.001
list.of.parameters <- seq(from = 10, to = 10000, by = 50)

#### ROUNDING
true.pos.rounding <- matrix(NA, nrow = length(list.of.parameters), ncol = 1)
sample.mean.rounding = membershipQuery(new.samp, rounding = TRUE, R = 10)

i = 1
for(parameters in list.of.parameters){
  cat("\nin rounding loop, number of parameters is now ", parameters)
  sample.mean <- sample.mean.rounding[1:parameters]
  pop.prob <- population.mean[1:parameters]
  calc.variance <- sum((sample.mean)^2 * (1-(pop.prob)^2))
  crit.val <- qnorm(my.alpha, mean = 0, sd = sqrt(calc.variance), lower.tail = FALSE, log.p = F

  history.rounding <- matrix(NA, nrow = 100, ncol = 2)
  # print(calc.variance, crit.val, nullDist.Dwork$criticalVal)
  for(row in 1:100){
    ## get real Alice from the sample and scale her to {-1,+1}
    alice <- new.samp[row,1:parameters]
    alice[alice==0] <- -1

    ## get the test stat for Alice
    test.alice.Dwork <- test.Dwork(alice=alice, sample.mean=sample.mean, population.mean=pop.pr
    history.rounding[row,1]<-test.alice.Dwork
    history.rounding[row,2]<-test.alice.Dwork>crit.val
  }
  true.pos.rounding[i, 1] = sum(history.rounding[,2]/nrow(history.rounding))
  i = i + 1
}


### plots for rounding
rounding.for.plots = data.frame(list.of.parameters , true.pos.rounding)

p1 <- ggplot(rounding.for.plots, aes(x = list.of.parameters, y = true.pos.rounding)) +
  geom_point() +
  labs(x = "Number of attributes", y = 'True positive probability', title =
         'True positive probability for different number of attributes\n(Rounding sample means)
  theme(plot.title = element_text(hjust = 0.5))

plot.rounding = grid.arrange(p1,  nrow = 1)
ggsave(filename = 'q3_rounding_R_10.pdf', plot = plot.rounding, width = 11, height = 5, units =



### Gaussian noise
```

```
true.pos.gaussian <- matrix(NA, nrow = length(list.of.parameters), ncol = 1)
sample.mean.gaussian = membershipQuery(new.samp, gaussian = TRUE, gaussian.sigma = 45)

i = 1
for(parameters in list.of.parameters){

  cat("\nin rounding loop, number of parameters is now ", parameters)
  sample.mean <- sample.mean.gaussian[1:parameters]
  pop.prob <- population.mean[1:parameters]
  calc.variance <- sum((sample.mean)^2 * (1-(pop.prob)^2))
  crit.val <- qnorm( my.alpha, mean = 0, sd = sqrt(calc.variance), lower.tail = FALSE, log.p =

  history.gaussian <- matrix(NA, nrow = 100, ncol = 2)

  for(row in 1:100){
    ## get real Alice from the sample
    alice <- new.samp[row,1:parameters]
    alice[alice==0] <- -1

    ## get the test stat for Alice
    test.alice.Dwork <- test.Dwork(alice=alice, sample.mean=sample.mean, population.mean=pop.pr
    history.gaussian[row,1]<-test.alice.Dwork
    history.gaussian[row,2]<-test.alice.Dwork>crit.val
  }
  true.pos.gaussian[i, 1] = sum(history.gaussian[,2]/nrow(history.gaussian))
  i = i + 1
}

## gaussian plots
gaus.for.plots = data.frame(list.of.parameters , true.pos.gaussian)

p2 <- ggplot(gaus.for.plots, aes(x = list.of.parameters, y = true.pos.gaussian)) +
  geom_point() +
  labs(x = "Number of attributes", y = 'True Positive Probability', title =
         'True positive probability for different number of attributes\nGaussian noise added to
  theme(plot.title = element_text(hjust = 0.5))

plots.gaus = grid.arrange(p2,  nrow = 1)

ggsave(filename = 'q3_gaussian_sigma_45.pdf', plot = plots.gaus, width = 11, height = 5, units


### Subsampling

true.pos.subsamp <- matrix(NA, nrow = length(list.of.parameters), ncol = 1)
sample.mean.subsamp = membershipQuery(new.samp, subsample = TRUE, subsample.size = 50)

i = 1
```

```
for(parameters in list.of.parameters){

  cat("\nin rounding loop, number of parameters is now ", parameters)

  sample.mean <- sample.mean.subsamp[1:parameters]
  pop.prob <- population.mean[1:parameters]
  calc.variance <- sum((sample.mean)^2 * (1-(pop.prob)^2))
  crit.val <- qnorm(my.alpha, mean = 0, sd = sqrt(calc.variance), lower.tail = FALSE, log.p = F

  history.subsamp <- matrix(NA, nrow = 100, ncol = 2)

  for(row in 1:100){
    ## get real Alice from the sample
    alice <- new.samp[row,1:parameters]
    alice[alice==0] <- -1

    ## get the test stat for Alice
    test.alice.Dwork <- test.Dwork(alice=alice, sample.mean=sample.mean, population.mean=pop.pr

    history.subsamp[row,1]<-test.alice.Dwork
    history.subsamp[row,2]<-test.alice.Dwork>crit.val
  }
  true.pos.subsamp[i, 1] = sum(history.subsamp[,2]/nrow(history.subsamp))
  i = i+1
}



## subsampling plots
subsamp.for.plots = data.frame(list.of.parameters , true.pos.subsamp)

p3 <- ggplot(subsamp.for.plots, aes(x = list.of.parameters, y = true.pos.subsamp)) +
  geom_point() +
  labs(x = "Number of attributes", y = 'True Positive Probability', title =
         'True positive probability for different number of attributes\nsubsampling') +
  theme(plot.title = element_text(hjust = 0.5))

plots.subsamp = grid.arrange(p3,  nrow = 1)
ggsave(filename = 'q3_subsamp_t_50.pdf', plot = plots.gaus, width = 11, height = 5, units = 'in
```