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
```