Untitled

Albert Bertran

2023-05-25

install.packages("LaplacesDemon")

adult\_train <- read.csv("D:/adult\_train.csv")  
library(LaplacesDemon)

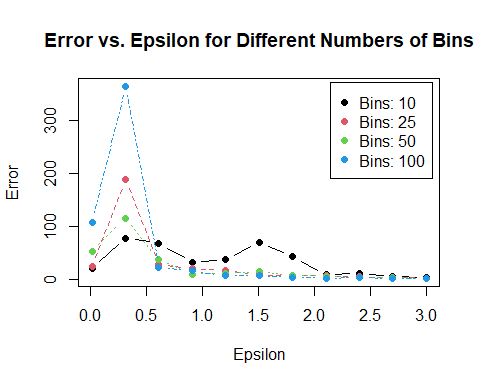
## Warning: package 'LaplacesDemon' was built under R version 4.2.3

adult <- adult\_train[c('age')]

laplace\_mechanism1 <- function(x, epsilon) {  
 sensitivity <- 125  
 scale <- sensitivity / epsilon  
 noise <- rlaplace(n = 1, loc = 0, scale = scale)  
 return(x + noise)  
}  
  
dp\_naive <- function(adult, epsilon) {  
 n <- nrow(adult)  
 d <- ncol(adult)  
 dp\_data <- matrix(0, nrow = n, ncol = d)  
 for (i in 1:n) {  
 dp\_data[i] <- laplace\_mechanism1(adult[i,], epsilon)  
 }  
 return(as.data.frame(dp\_data))  
}  
  
epsilon <- 0.01  
errors = c()  
meanAfterDp = c()  
for (i in 1:10) {  
 histogramE = c()  
 histogramM = c()  
 for (j in 1:25) {  
 dp\_data <- dp\_naive(adult, epsilon)  
 mdp <- mean(dp\_data$V1)  
 morig <- mean(adult$age)  
 histogramE[j] <- abs(mdp - morig)  
 histogramM[j] <- mdp  
 }  
 e <- mean(histogramE)  
 errors[i] <- e  
 m <- mean(histogramM)  
 meanAfterDp[i] <- m  
 epsilon <- epsilon + 0.299  
}  
errors

## [1] 80.3474908 2.5884278 1.2637386 0.8817059 0.6774335 0.5414435  
## [7] 0.4115679 0.3811028 0.3883267 0.3035289

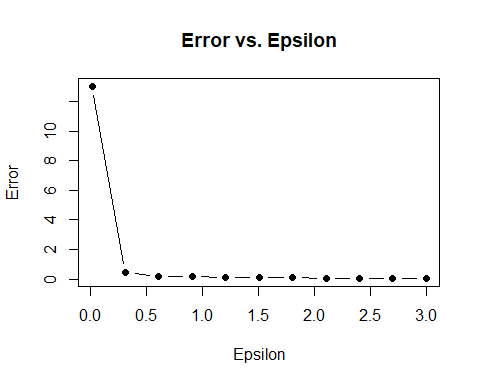
dp\_histogram\_mean <- function(adult, epsilon, num\_bins) {  
 n <- length(adult$age)  
   
 # Calculate histogram properties  
 min\_age <- min(adult$age)  
 max\_age <- max(adult$age)  
 bin\_width <- (max\_age - min\_age) / num\_bins  
   
 # Calculate global sensitivity of the histogram query  
 max\_count <- 0  
 histogram\_counts <- rep(0, num\_bins)  
   
 for (i in 1:n) {  
 individual\_age <- adult$age[i]  
 bin\_index <- pmin(pmax(floor((individual\_age - min\_age) / bin\_width) + 1, 1), num\_bins)  
 histogram\_counts[bin\_index] <- histogram\_counts[bin\_index] + 1  
 max\_count <- max(max\_count, histogram\_counts[bin\_index])  
 }  
   
 # Apply Laplace noise to protect the histogram counts  
 noisy\_counts <- histogram\_counts + rlaplace(num\_bins, scale = max\_count / epsilon)  
   
 # Compute the differentially private mean from the protected histogram  
 sum\_age <- 0  
 total\_count <- sum(noisy\_counts)  
   
 for (i in 1:num\_bins) {  
 bin\_age <- min\_age + (i - 0.5) \* bin\_width  
 sum\_age <- sum\_age + noisy\_counts[i] \* bin\_age  
 }  
   
 dp\_mean <- sum\_age / total\_count  
   
 return(dp\_mean)  
}  
  
# Main code  
epsilon\_values <- seq(0.01, 3, by = 0.299)  
num\_bins\_list <- c(10, 25, 50, 100)  
num\_iterations <- 25  
  
mean\_values <- matrix(0, nrow = length(epsilon\_values), ncol = length(num\_bins\_list))  
error\_values <- matrix(0, nrow = length(epsilon\_values), ncol = length(num\_bins\_list))  
  
for (i in 1:length(epsilon\_values)) {  
 epsilon <- epsilon\_values[i]  
   
 for (j in 1:length(num\_bins\_list)) {  
 num\_bins <- num\_bins\_list[j]  
   
 histogram\_errors <- matrix(0, nrow = num\_iterations, ncol = 1)  
 mean\_after\_dp\_list <- matrix(0, nrow = num\_iterations, ncol = 1)  
   
 for (iter in 1:num\_iterations) {  
 dp\_mean <- dp\_histogram\_mean(adult, epsilon, num\_bins)  
 mean\_after\_dp\_list[iter] <- dp\_mean  
 orig\_mean <- mean(adult$age)  
 histogram\_errors[iter] <- abs(dp\_mean - orig\_mean)  
 }  
   
 error\_values[i, j] <- mean(histogram\_errors)  
 mean\_values[i, j] <- mean(mean\_after\_dp\_list)  
 }  
}  
  
# Plot the results  
plot\_labels <- paste("Bins:", num\_bins\_list)  
  
matplot(epsilon\_values, error\_values, type = "b", pch = 16, xlab = "Epsilon", ylab = "Error",  
 main = "Error vs. Epsilon for Different Numbers of Bins", col = 1:length(num\_bins\_list))  
legend("topright", legend = plot\_labels, col = 1:length(num\_bins\_list), pch = 16, inset = 0.02)



print(mean(adult$age))

## [1] 38.58165

# Laplace mechanism for differentially private summation query  
dp\_summation <- function(data, epsilon) {  
 sensitivity <- max(data) - min(data)  
 scale <- sensitivity / epsilon  
 noise <- rlaplace(n = 1, loc = 0, scale = scale)  
 return(sum(data) + noise)  
}  
  
# Laplace mechanism for differentially private counting query  
dp\_counting <- function(data, epsilon) {  
 sensitivity <- 125  
 scale <- sensitivity / epsilon  
 noise <- rlaplace(n = 1, loc = 0, scale = scale)  
 return(length(data) + noise)  
}  
  
# Initialize vectors to store error and mean values  
epsilon\_values <- seq(0.01, 3, by = 0.299)  
error\_values <- rep(0, length(epsilon\_values))  
mean\_values <- rep(0, length(epsilon\_values))  
  
# Iterate over epsilon values  
for (i in 1:length(epsilon\_values)) {  
 epsilon <- epsilon\_values[i]  
 mean\_results <- numeric(25)  
 error\_results <- numeric(25)  
   
 # Compute mean and error 25 times  
 for (j in 1:25) {  
 dp\_sum <- dp\_summation(adult$age, epsilon)  
 dp\_count <- dp\_counting(adult$age, epsilon)  
 orig\_mean <- mean(adult$age)  
   
 dp\_mean <- dp\_sum / dp\_count  
 mean\_results[j] <- dp\_mean  
 error\_results[j] <- abs(dp\_mean - orig\_mean)  
 }  
   
 # Store mean and error values  
 mean\_values[i] <- mean(mean\_results)  
 error\_values[i] <- mean(error\_results)  
}  
  
# Plot the results  
plot(epsilon\_values, error\_values, type = "b", pch = 16, xlab = "Epsilon", ylab = "Error", main = "Error vs. Epsilon")



plot(epsilon\_values, mean\_values, type = "b", pch = 16, xlab = "Epsilon", ylab = "Mean", main = "Mean vs. Epsilon")

