Differentially Private Verification Methods on Prediction Models

Haoyang Yu

10/10/2017

```
library(rmutil)
```

```
## Warning: package 'rmutil' was built under R version 3.3.2
```

```
##
## Attaching package: 'rmutil'
```

```
## The following object is masked from 'package:stats':
##
##     nobs
```

```
################################
#Explanatory variables Generator
################################

rnorm.10<-function(n,p){
  m=p*n
  x.1<-matrix(rnorm(m/10,mean=0),n/10,p)
  x.2<-matrix(rnorm(m/10,mean=1),n/10,p)
  x.3<-matrix(rnorm(m/10,mean=2),n/10,p)
  x.4<-matrix(rnorm(m/10,mean=3),n/10,p)
  x.5<-matrix(rnorm(m/10,mean=4),n/10,p)
  x.6<-matrix(rnorm(m/10,mean=5),n/10,p)
  x.7<-matrix(rnorm(m/10,mean=6),n/10,p)
  x.8<-matrix(rnorm(m/10,mean=7),n/10,p)
  x.9<-matrix(rnorm(m/10,mean=8),n/10,p)
  x.10<-matrix(rnorm(m/10,mean=9),n/10,p)
  x<-rbind(x.1,x.2,x.3,x.4,x.5,x.6,x.7,x.8,x.9,x.10)
  return(x)
}
```

Method 1. Prediction Tolerance Interval

```
################################
#Prediction Tolerance Interval
######95% Confidence Intervals
################################

CI.function<-function(fit){
  r=range(Authentic.Data$y,upperl,lowerl)
  #plot the 95% confidence intervals
  plot(NULL,xlim=r,ylim=r,ylab="Authentic response",
       xlab="Authentic data points",main="95% Confidence Intervals")
```

```
  for(i in 1:n)
  segments(Authentic.Data$y[i],upperl[i],
           Authentic.Data$y[i],lowerl[i],col="gray")
  points(Authentic.Data$y,Authentic.Data$y,cex=0.5)
  S=sum((lowerl<Authentic.Data$y) & (upperl>Authentic.Data$y))
  S.star = S + round(rlaplace(1,0,1))
  P.star=min(1,S.star/n)
  sprintf("S is %s; S.star is %s; P.Star is %s",S,S.star,P.star)
}
```

Method 2. Pnorm Histogram

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function<-function(fit){
  p<-vector()
  for (i in 1:n){
    p[i]<-pnorm(Authentic.Data$y[i],Mean[i],summary(fit)$sigma)
  }
  ## Get the Frequency of the intervals
  hist_p<-hist(p,breaks=seq(0,1,by=0.1),plot=FALSE)$counts

  hist_r<-vector()
  ## Add Laplace Noise to the Frequency
  for (j in 1:10){
  hist_r[j]<-round(rlaplace(1,0,1))
}
  hist_n<-hist_p+hist_r+10
  counts<-as.matrix(rbind(hist_p,hist_r))
  xx<-barplot(counts,main="Pnorm Histogram",
              names.arg=c("1","2","3","4","5","6","7","8","9","10"),
              xlab = "Intervals from [0,0.1] to [0.9,1]",
              ylab = "Frequency")
  # This is the Pnorm Histogram to release in the second algorithm
}
```

Method 3: KS test

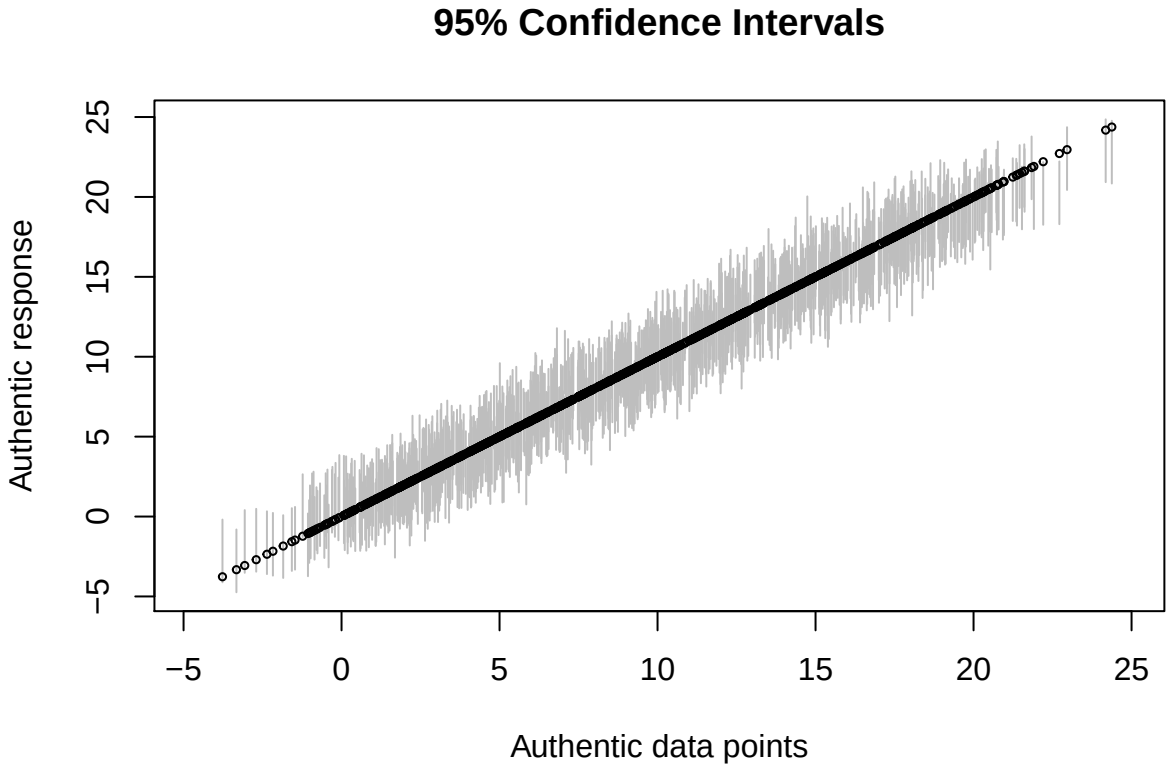
```
ks.function<-function(Mean,fit,x){
       ## making predicitons centered at the predicated mean
       ## with estmated standard errors
       m <- length(Mean)
       new.PI<-vector()
       for (i in 1:m){
         new.PI[i]=rnorm(1,mean=Mean[i],sd=summary(fit)$sigma)
       }

       ## Calculate the KS statistics with Laplace noise
       y <- new.PI

       ## Based on the sample sizes of the two data sets
       ## Get the P value caculation function
```

```
x <- x[!is.na(x)]
n <- length(x)
y <- y[!is.na(y)]
n.x <- as.double(n)             # to avoid integer overflow
n.y <- length(y)
n <- n.x * n.y / (n.x + n.y)
alternative = "two.sided"
pkstwo <- function(x, tol = 1e-6) {
     ## Compute \sum_{-\infty}^\infty (-1)^k e^{-2k^2x^2}
     ## Not really needed at this generality for computing a single
     ## asymptotic p-value as below.
     if(is.numeric(x)) x <- as.double(x)
     else stop("argument 'x' must be numeric")
     p <- rep(0, length(x))
     p[is.na(x)] <- NA
     IND <- which(!is.na(x) & (x > 0))
     if(length(IND)) p[IND] <- .Call(stats:::C_pKS2, p = x[IND], tol)
     p
 }
 ## <FIXME>
 ## Currently, p-values for the two-sided two-sample case are
 ## exact if n.x * n.y < 10000 (unless controlled explicitly).
 ## In all other cases, the asymptotic distribution is used
 ## directly.  But: let m and n be the min and max of the sample
 ## sizes, respectively.  Then, according to Kim and Jennrich
 ## (1973), if m < n/10, we should use the
 ## * Kolmogorov approximation with c.c. -1/(2*n) if 1 < m < 80;
 ## * Smirnov approximation with c.c. 1/(2*sqrt(n)) if m >= 80.

 PVAL <- function(STATISTIC){
   p <- ifelse(alternative == "two.sided",
                1 - pkstwo(sqrt(n) * STATISTIC),
                exp(- 2 * n * STATISTIC^2))
   p <- min(1.0, max(0.0, p))
   return(p)
   }

 ## Create 10001 grids from 0 to 1
 ks.grid<-seq(0,1,0.0001)

 ## Caculate the P value for each grid
 PVAL.grid <- sapply(ks.grid, PVAL)

 ## Caculate the probability (discrete pdf) for each grid
 ## By substracting the next grid's Pvalue from the previous P value
 ## Since the P value decrease when the KS statistics increase
 pdf.grid <- vector()
 for (i in 1:length(PVAL.grid)-1){
   pdf.grid[i] <- max(PVAL.grid[i]-PVAL.grid[i+1],0)
 }
 pdf.grid[length(ks.grid)] = 0
 sample.time=length(which(pdf.grid!=0))
```

```
        ## Create sampling indicators
        ## Sample a value of the KS quantity.
        ## Then add Laplace noise to it, replace, save
        ## Repeat.
        ks.laplace <- vector()
        for(j in 1:10000){
            ks.laplace[j] <- sample(ks.grid, size = 1, prob = pdf.grid) +
              rlaplace(1,0,s=2/m)
            ks.laplace[j] <- max(0,ks.laplace[j])
            ks.laplace[j] <- min(1,ks.laplace[j])
        }
        ks.laplace<-sort(ks.laplace)
        ## Calculate the KS statistics with Laplace noise
        ks <- ks.test(x,y)
        ks.star <- ks$statistic+rlaplace(1,0,2/m)

        ## Find the P value for the noised KS statistics
        ## In the reference distribution
        rank.indicator <- 0
        pval.star = 1
        for (i in 2:length(ks.laplace)) {
          if(ks.star >= ks.laplace[i-1] && ks.star < ks.laplace[i]){
            pval.star <- 1-(i-1)/10000
          }
          if(ks.star>ks.laplace[i]){
            pval.star <- 0
          }
        }

        sprintf("Differentially Private: KS.Star is %s; Pvalue.star is %s",
                ks.star,pval.star)
}

ks.actual<-function(Mean,fit,x){
        ## making predicitons centered at the predicated mean
        ## with estmated standard errors
        m <- length(Mean)
        new.PI<-vector()
        for (i in 1:m){
          new.PI[i]=rnorm(1,mean=Mean[i],sd=summary(fit)$sigma)}
        y <- new.PI
        ks <- ks.test(x,y)
        ks.statistics <- ks$statistic
        ks.pvalue <- ks$p.value
        sprintf("Actual KS test KS.statistics is %s; KS.Pvalue is %s",
                ks.statistics,ks.pvalue)
}
```

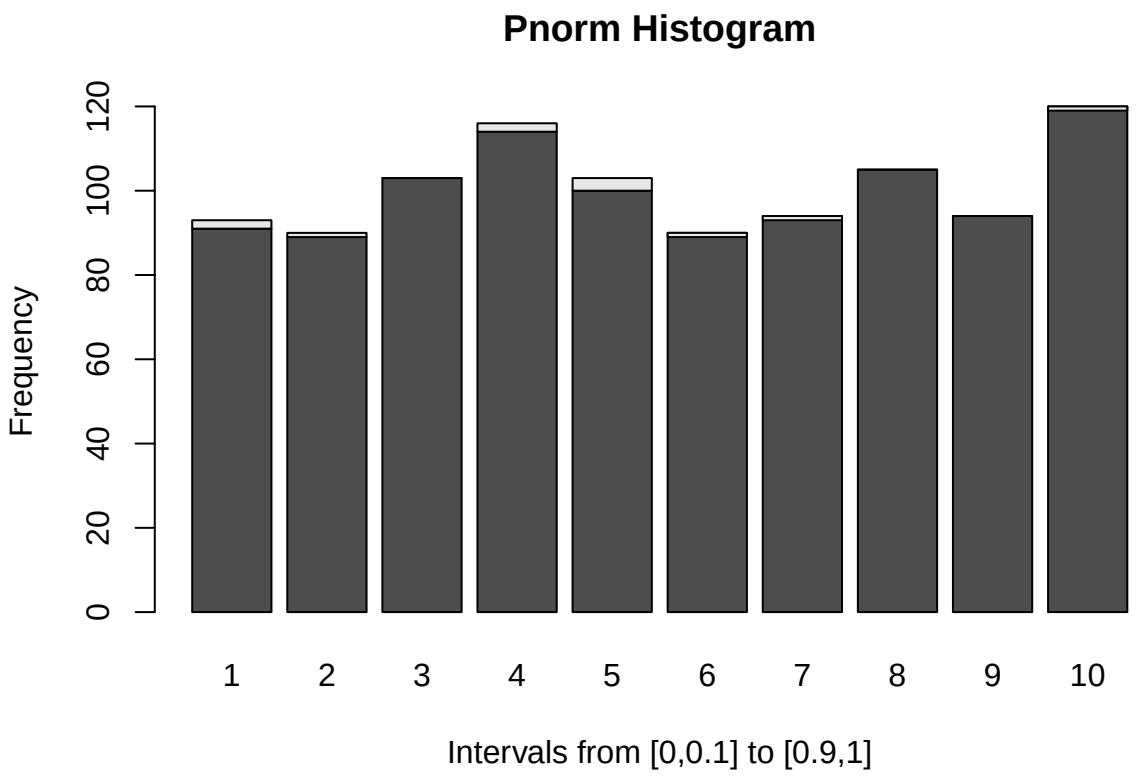
4.1 Similar Data Generater 4.1.1 Same Generator & Perfect Model
Authentic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\sigma}$

```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
# predictors
Y = 1+ X[,1]+X[,2]+rnorm(n,0,sigma)  # response variable
Authentic.Data = data.frame(y=as.vector(Y),x=X)

#########################
# Generate synthetic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
aut.fit = lm(y~1+x.1+x.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X[,1]+aut.cof[3]*X[,2]+
  rnorm(n,0,aut.sig)  # response variable

Synthetic.Data = data.frame(y=as.vector(Y),x=X)


##############################
# Fit model to synthetic data
##############################
fit = lm(y~1+x.1+x.2,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################

PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
lowerl=PI[,2]
upperl=PI[,3]

##############################
#Prediction Tolerance Interval
######95% Confidence Intervals
##############################
CI.function(fit)
```




```
## [1] "S is 948; S.star is 951; P.Star is 0.951"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm


########################
####### KS test Methods
########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.0263433534593942; Pvalue.star is 0.8675"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.02; KS.Pvalue is 0.988261077643524"
```

```
#This is the actual KS test result
```

4.1.2.1
Authentic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\sigma}$

```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
```

```r
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
Y = 1+ X[,1]+X[,2] +rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X)
head(Authentic.Data)
```

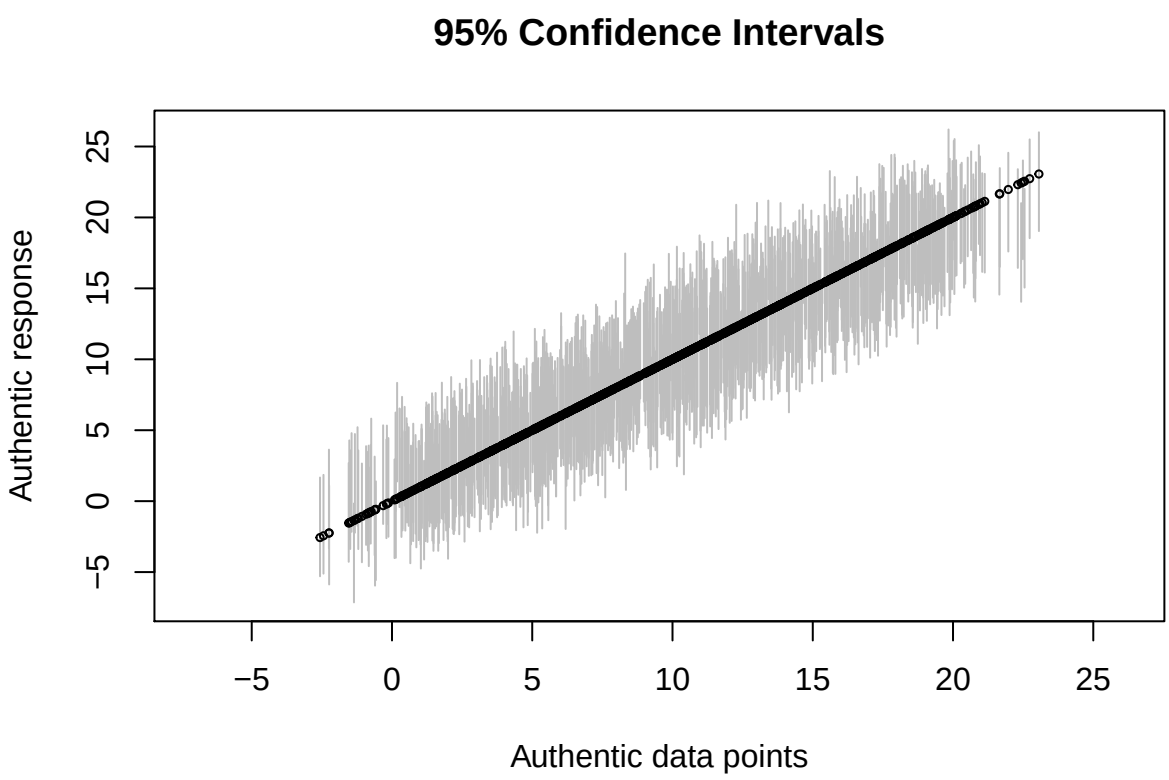
```
##             y        x.1        x.2
## 1  0.09036977 -1.0714703  0.5335423
## 2 -0.57625745 -1.8900888  0.4267164
## 3 -0.92273155 -0.5783600 -0.5637187
## 4  1.82137633  0.3732450 -0.2862789
## 5  2.22708967  1.4770242  0.1669693
## 6  0.51988167  0.4656205 -0.4412184
```

```r
##########################
# Generate synthetic data
##########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
aut.fit = lm(y~1+x.1+x.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X[,1]+aut.cof[3]*X[,2]+
  rnorm(n,0,aut.sig)  # response variable

Synthetic.Data = data.frame(y=as.vector(Y),x=X)
head(Synthetic.Data)
```

```
##             y        x.1         x.2
## 1  1.70595626  1.9011558 -0.04406127
## 2  0.77440329 -0.5548186  1.28015300
## 3  0.87686373  0.8713014 -0.64150597
## 4  0.03977658 -1.4551240  0.07912498
## 5  0.09914159 -0.8718381  1.32415161
## 6 -0.55945766 -0.7053956 -0.72181598
```

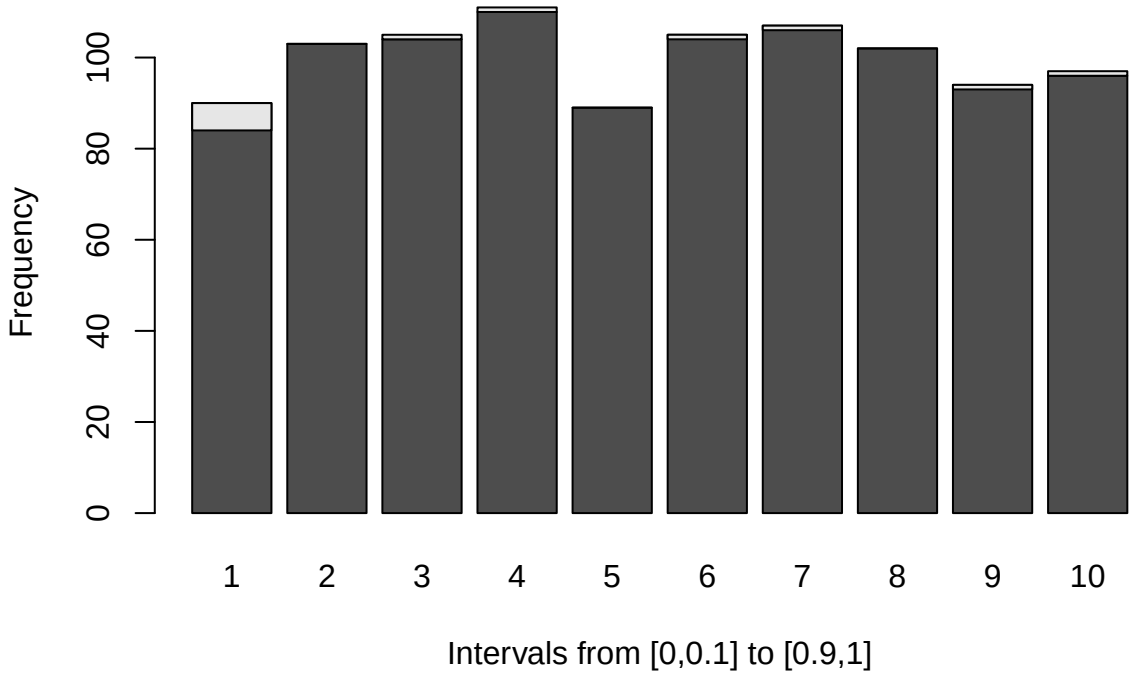
```r
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.1,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################


PI=predict(fit,newdata = Authentic.Data,interval='prediction')
```

```
Mean=PI[,1]
lowerl=PI[,2]
upperl=PI[,3]

##############################
#Prediction Tolerance Interval
######95% Confidence Intervals
##############################
CI.function(fit)
```




```
## [1] "S is 956; S.star is 958; P.Star is 0.958"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

#########################
#### P Values for KS Test
#########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.0167596472694566; Pvalue.star is 0.9954"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.032; KS.Pvalue is 0.685231404257084"
```

```
#This is the actual KS test result
```

4.1.2.2
Authentic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1^2 + \hat{\beta}_1 x_2^2 + \hat{\sigma}$

```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
```

```
X = rnorm.10(n,p) # predictors
X.S=X^2
Y = 1+ X[,1]+X[,2] +rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Authentic.Data)
```

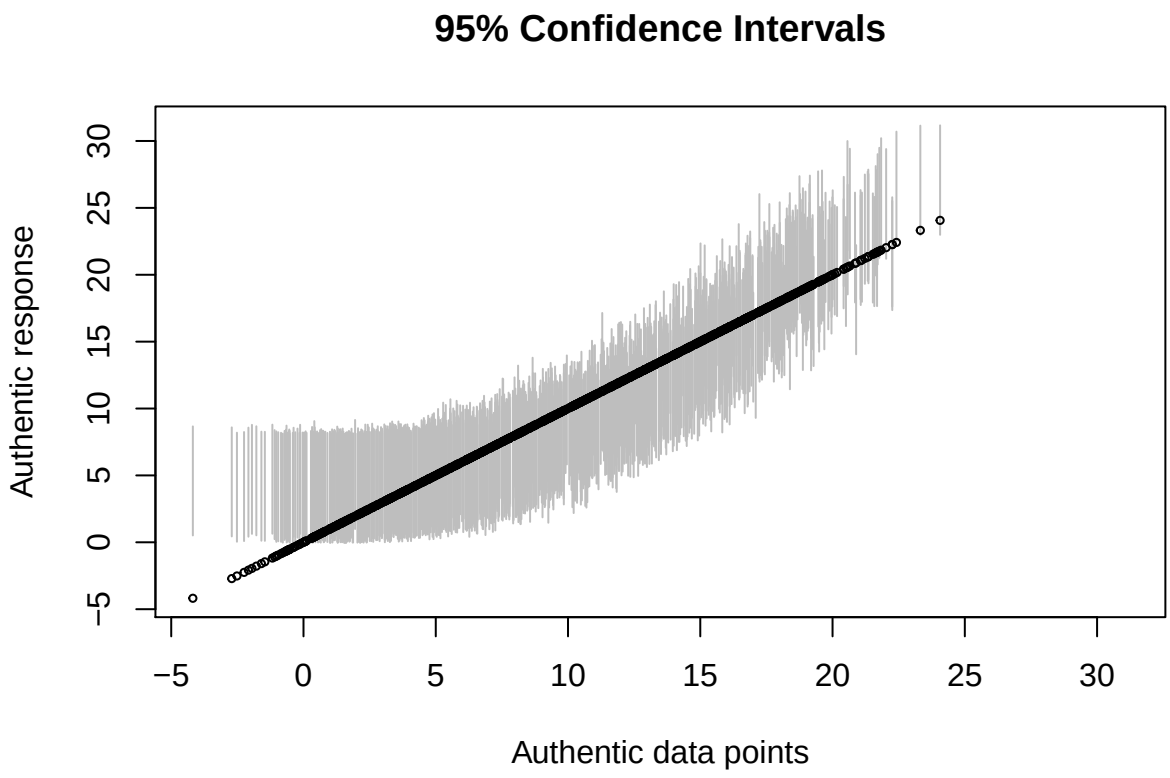
```
##           y         x.1        x.2       x.s.1      x.s.2
## 1 3.6012419  0.99330639  0.1337288 0.986657579 0.01788339
## 2 3.3074072  0.78164312  0.9903257 0.610965972 0.98074495
## 3 0.3481654 -0.04582446  0.3213500 0.002099881 0.10326580
## 4 2.9621888  1.32752140  1.3413500 1.762313064 1.79921976
## 5 0.7755226  0.95398940 -0.6750890 0.910095779 0.45574510
## 6 1.4783376 -1.14335496  0.6474006 1.307260559 0.41912756
```

```
##########################
# Generate synthetic data
##########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
X.S = X^2
aut.fit = lm(y~1+x.1+x.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X[,1]+aut.cof[3]*X[,2]+
  rnorm(n,0,aut.sig)  # response variable

Synthetic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Synthetic.Data)
```

```
##            y         x.1        x.2       x.s.1      x.s.2
## 1  0.3544153  0.26809854  0.7162338 0.071876825 0.51299093
## 2 -0.2505219 -1.16499065 -0.5499923 1.357203214 0.30249149
## 3 -0.7302675  0.07818918 -0.2292682 0.006113547 0.05256393
## 4  1.3426276 -1.24626720  0.7624886 1.553181929 0.58138888
## 5  1.9165191  0.13982210  0.2744481 0.019550219 0.07532178
## 6  0.2075701  0.75368993 -1.0980099 0.568048511 1.20562571
```

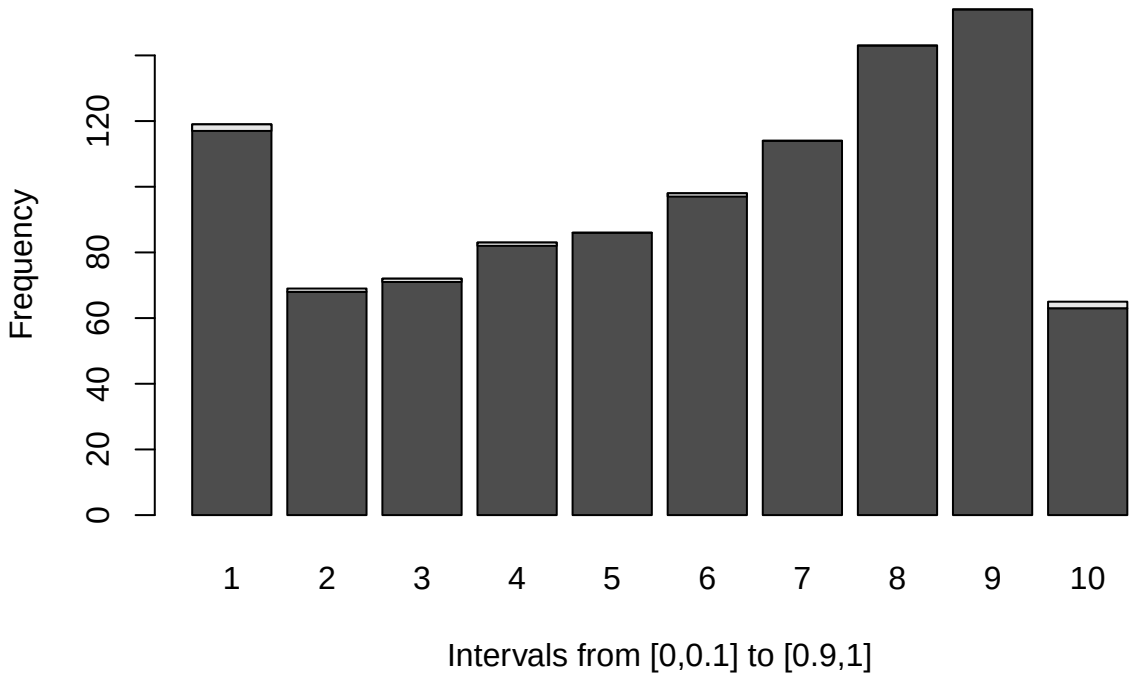
```
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.s.1+x.s.2,data=Synthetic.Data)

####################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
####################################
```

```
PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
lowerl=PI[,2]
upperl=PI[,3]

##############################
#Prediction Tolerance Interval
######95% Confidence Intervals
##############################
CI.function(fit)
```




```
## [1] "S is 954; S.star is 954; P.Star is 0.954"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

#########################
#### P Values for KS Test
#########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.105931644709866; Pvalue.star is 0"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.084; KS.Pvalue is 0.00172444017038353"
```

```
#This is the actual KS test result
```

4.1.3.1
Authentic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1^2 + \hat{\sigma}$

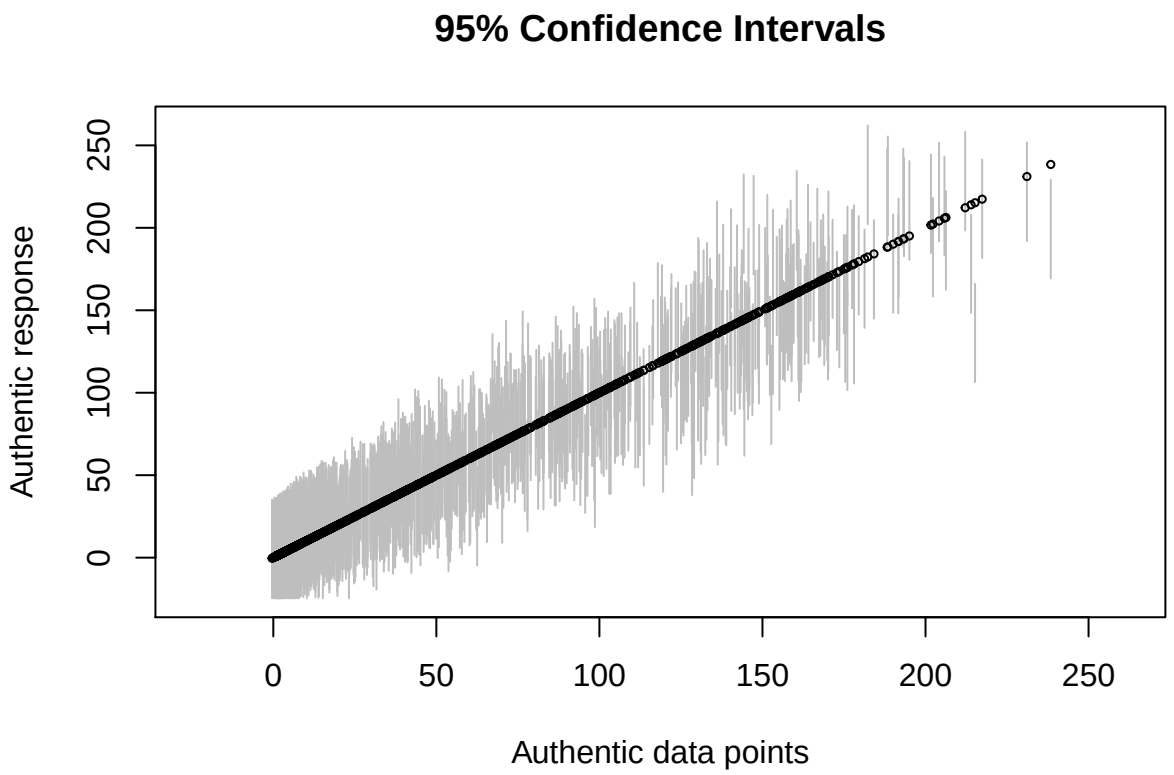
```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
```

```
X = rnorm.10(n,p) # predictors
X.S=X^2
Y = 1+ X.S[,1]+X.S[,2]+rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)

#########################
# Generate synthetic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
X.S=X^2
aut.fit = lm(y~1+x.s.1+x.s.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X.S[,1]+aut.cof[3]*X.S[,2]+
  rnorm(n,0,aut.sig)  # response variable

Synthetic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Synthetic.Data)
```

```
##           y        x.1        x.2      x.s.1      x.s.2
## 1 0.4180337 -0.1998278 -0.0799553 0.03993114 0.00639285
## 2 3.6420067 -0.9542585 -0.9867654 0.91060924 0.97370589
## 3 0.7077446 -0.8114180 -0.6165847 0.65839915 0.38017665
## 4 1.7235999 -0.1658969  0.2657166 0.02752179 0.07060532
## 5 1.7308109  0.4527126  0.1300069 0.20494874 0.01690178
## 6 7.2983410 -2.2993474 -0.8594619 5.28699863 0.73867471
```

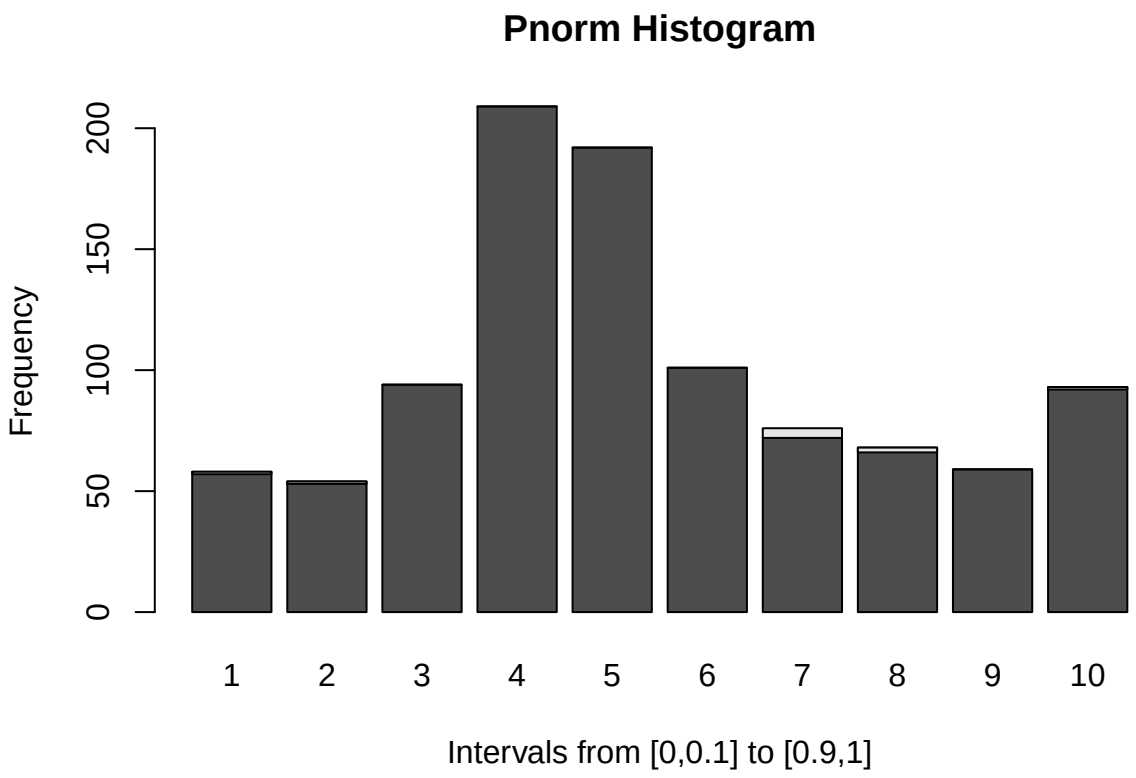
```
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.s.1,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################


PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
lowerl=PI[,2]
upperl=PI[,3]

###############################
#Prediction Tolerance Interval
######95% Confidence Intervals
###############################
```

```
CI.function(fit)
```




```
## [1] "S is 943; S.star is 942; P.Star is 0.942"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

##########################
#### P Values for KS Test
##########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.095186767085168; Pvalue.star is 0.000499999999999945"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.093; KS.Pvalue is 0.000350604125194942"
```

```
#This is the actual KS test result
```

4.1.3.2
Authentic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\sigma}$

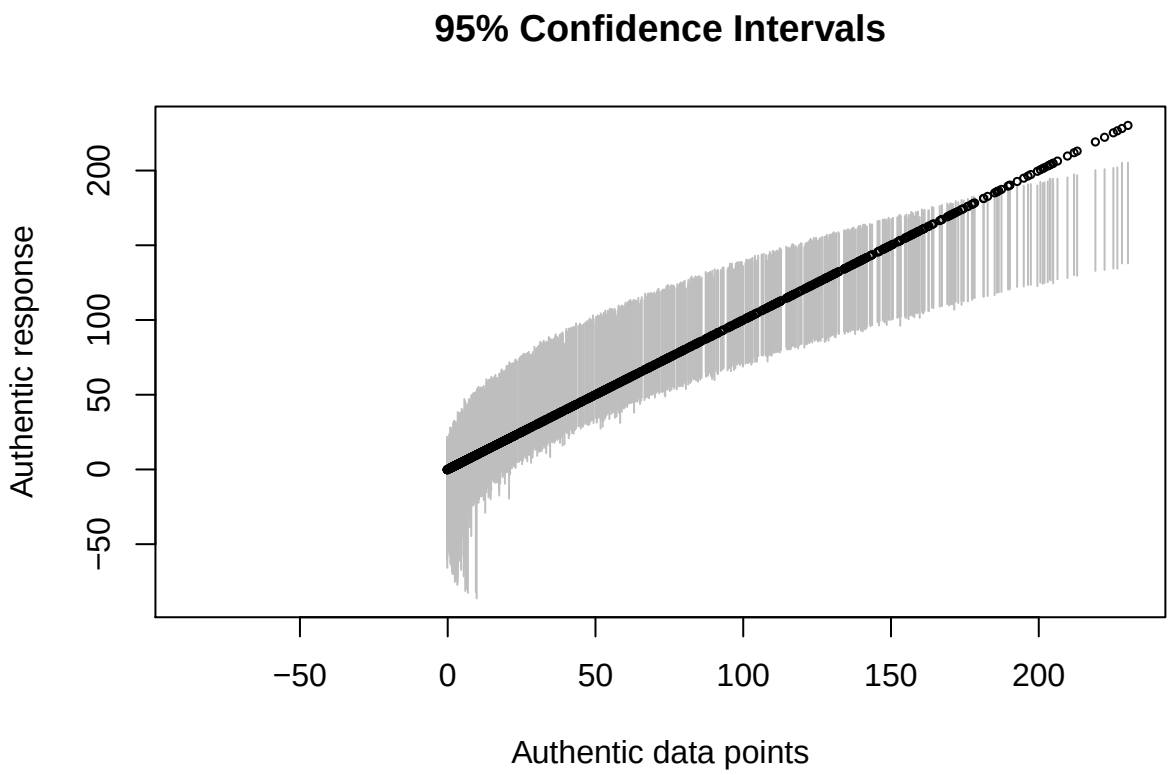
```
##########################
# Generate authentic data
##########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
```

```
X = rnorm.10(n,p) # predictors
X.S=X^2
Y = 1+ X.S[,1]+X.S[,2]+rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)

#########################
# Generate synthetic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
X.S=X^2
aut.fit = lm(y~1+x.s.1+x.s.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X.S[,1]+aut.cof[3]*X.S[,2]+
  rnorm(n,0,aut.sig)  # response variable

Synthetic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Synthetic.Data)
```

```
##           y        x.1        x.2       x.s.1     x.s.2
## 1 2.4321408  0.6331323  0.4372108 0.400856447 0.1911532
## 2 3.8207236 -1.4386811 -0.8215646 2.069803223 0.6749684
## 3 0.1267516  0.0480575  0.5298658 0.002309524 0.2807577
## 4 4.2623851 -0.9253613  1.5740557 0.856293598 2.4776513
## 5 8.0709716 -1.8219245  1.9308471 3.319408784 3.7281703
## 6 2.6118920 -0.3640377  1.2117835 0.132523468 1.4684192
```

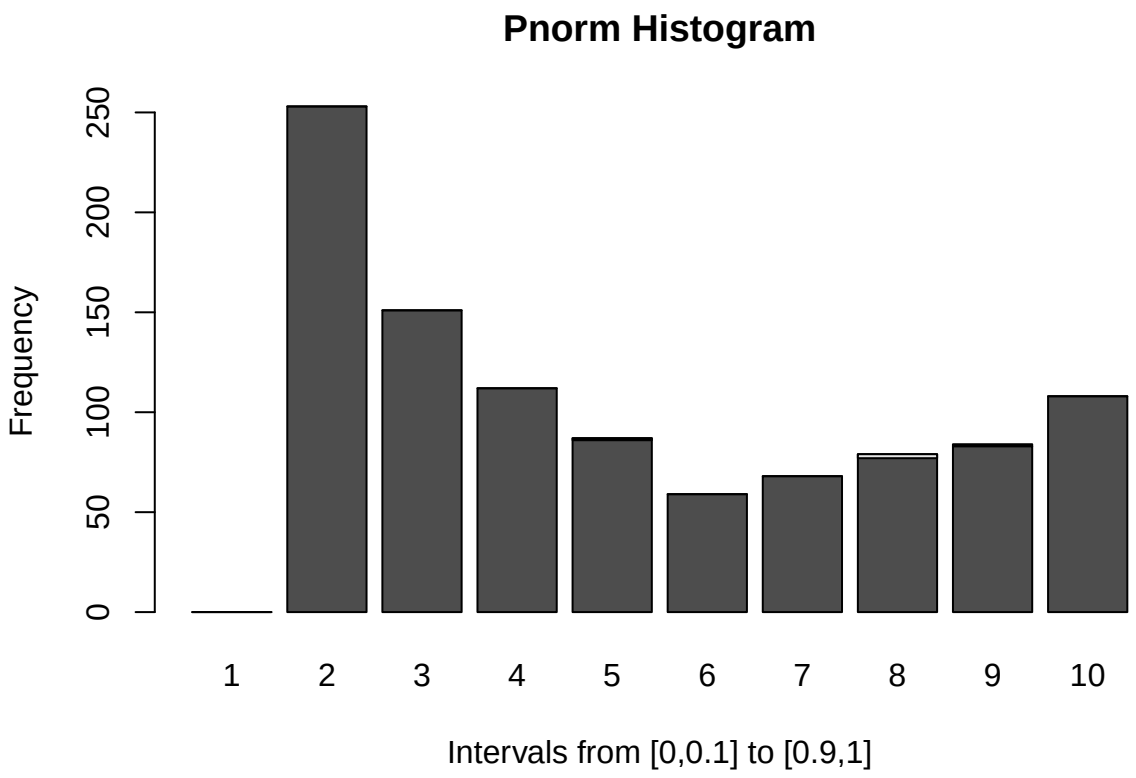
```
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.1+x.2,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################


PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
lowerl=PI[,2]
upperl=PI[,3]

###############################
#Prediction Tolerance Interval
######95% Confidence Intervals
###############################
```

```
CI.function(fit)
```




```
## [1] "S is 946; S.star is 945; P.Star is 0.945"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

#########################
#### P Values for KS Test
#########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.176860158974846; Pvalue.star is 0"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.17; KS.Pvalue is 5.62216939670179e-13"
```

```
#This is the actual KS test result
```

4.2.1.1 Different Data Generater & Correct Model
Authentic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\sigma}$

```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
```

```r
X = rnorm.10(n,p) # predictors
X.S = X^2
Y = 1+ X[,1] + X[,2] +rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Authentic.Data)
```

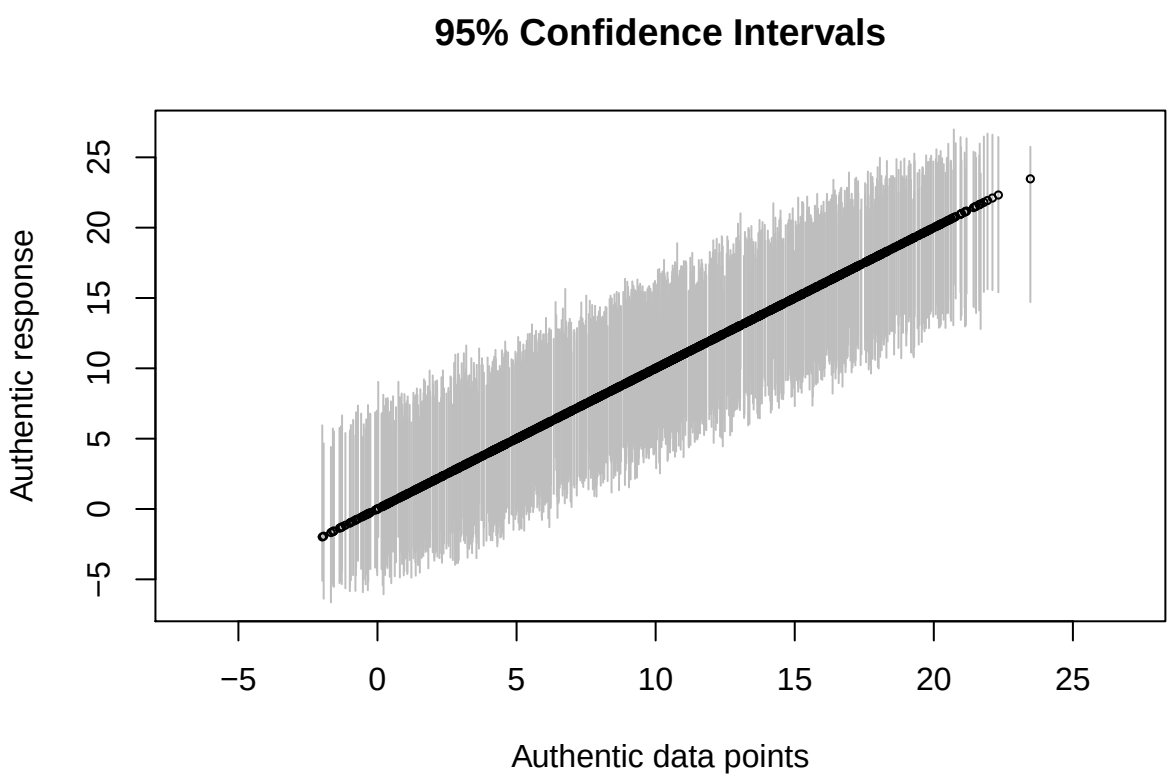
```
##            y        x.1        x.2      x.s.1     x.s.2
## 1  0.1465802  1.0991275 -1.3755823 1.20808137 1.8922268
## 2  0.3467923  0.5534757 -0.4880641 0.30633536 0.2382066
## 3  1.5940207  1.4169958 -0.7857884 2.00787719 0.6174634
## 4  0.3655324 -0.3114279  0.9817068 0.09698736 0.9637483
## 5 -1.9847649 -2.0223566  0.5175342 4.08992631 0.2678416
## 6  0.5603226  1.9815934 -0.8976416 3.92671245 0.8057604
```

```r
#########################
# Generate synthetic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
X.S=X^2
aut.fit = lm(y~1+x.s.1+x.s.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X.S[,1]+aut.cof[3]*X.S[,2]+
  rnorm(n,0,aut.sig) #response variable
Synthetic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Synthetic.Data)
```

```
##           y        x.1        x.2      x.s.1     x.s.2
## 1 2.6749601  1.6125439 -0.4618730 2.60029793 0.2133266
## 2 3.7126547 -2.6089544  0.6287206 6.80664290 0.3952896
## 3 4.7050118 -0.2290952 -0.5028015 0.05248461 0.2528093
## 4 2.7851608 -0.2180690 -0.6804105 0.04755409 0.4629585
## 5 5.9856339 -0.4100022  1.6947287 0.16810176 2.8721053
## 6 0.8537815 -1.5411770 -1.5753748 2.37522651 2.4818058
```

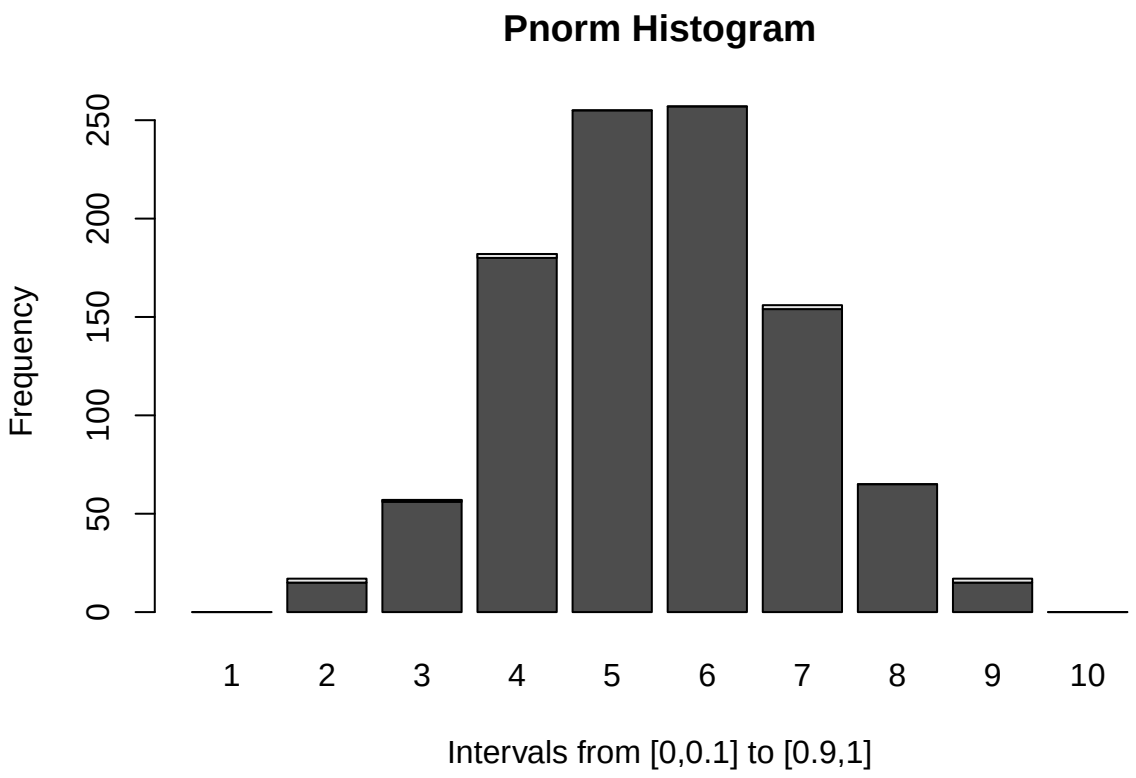
```r
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.1+x.2,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################

PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
```

```
lowerl=PI[,2]
upperl=PI[,3]

##############################
#Prediction Tolerance Interval
######95% Confidence Intervals
##############################

CI.function(fit)
```




```
## [1] "S is 1000; S.star is 1000; P.Star is 1"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

#########################
#### P Values for KS Test
#########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.0280471251111626; Pvalue.star is 0.8214"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.039; KS.Pvalue is 0.432432065694456"
```

```
#This is the actual KS test result
```

4.2.1.2 Different Data Generater & Correct Model
Authentic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1^2 + \hat{\beta}_2 x_2^2 + \hat{\sigma}$

```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
```

```
X = rnorm.10(n,p) # predictors
X.S = X^2
Y = 1+ X[,1] + X[,2] +rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Authentic.Data)
```

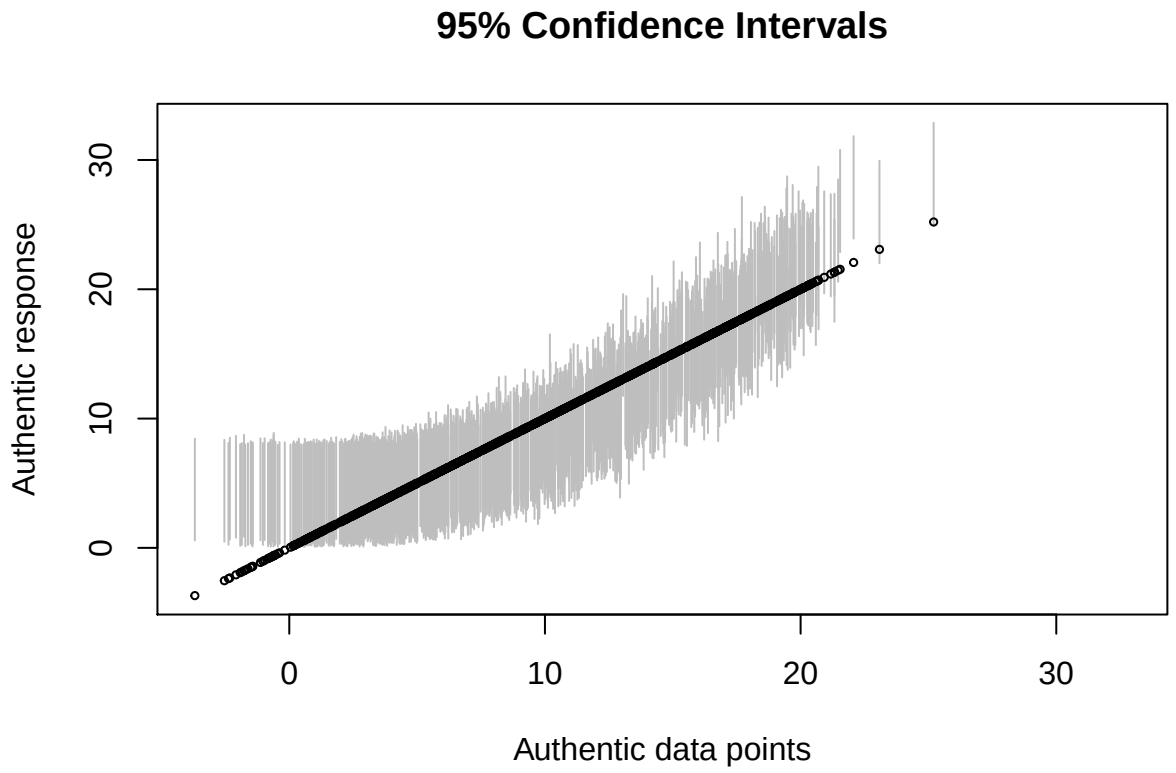
```
##            y        x.1         x.2     x.s.1       x.s.2
## 1  0.2189900 -0.4002122  0.81412427 0.1601698 0.662798319
## 2 -0.5643651 -1.0300574 -0.30952127 1.0610183 0.095803417
## 3  1.1019559 -0.3554045  0.38268672 0.1263124 0.146449129
## 4  1.2602248  1.3061906  0.26649346 1.7061338 0.071018765
## 5  1.2460683 -0.5886641 -0.01963031 0.3465255 0.000385349
## 6  1.0848551  0.1649415  0.53539693 0.0272057 0.286649876
```

```
#########################
# Generate synthetic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
X.S=X^2
aut.fit = lm(y~1+x.s.1+x.s.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X.S[,1]+aut.cof[3]*X.S[,2]+
  rnorm(n,0,aut.sig) #response variable
Synthetic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Synthetic.Data)
```

```
##          y        x.1        x.2     x.s.1      x.s.2
## 1 5.276140  0.4708139  0.2190943 0.2216658 0.04800233
## 2 1.680232 -0.8872957 -0.9520433 0.7872936 0.90638639
## 3 8.150549 -0.9947208 -0.1010931 0.9894694 0.01021982
## 4 3.483658 -0.8577660 -0.3312997 0.7357624 0.10975950
## 5 8.022264  0.6473239 -1.7337807 0.4190283 3.00599553
## 6 7.548697  2.7040285  1.7649990 7.3117704 3.11522148
```

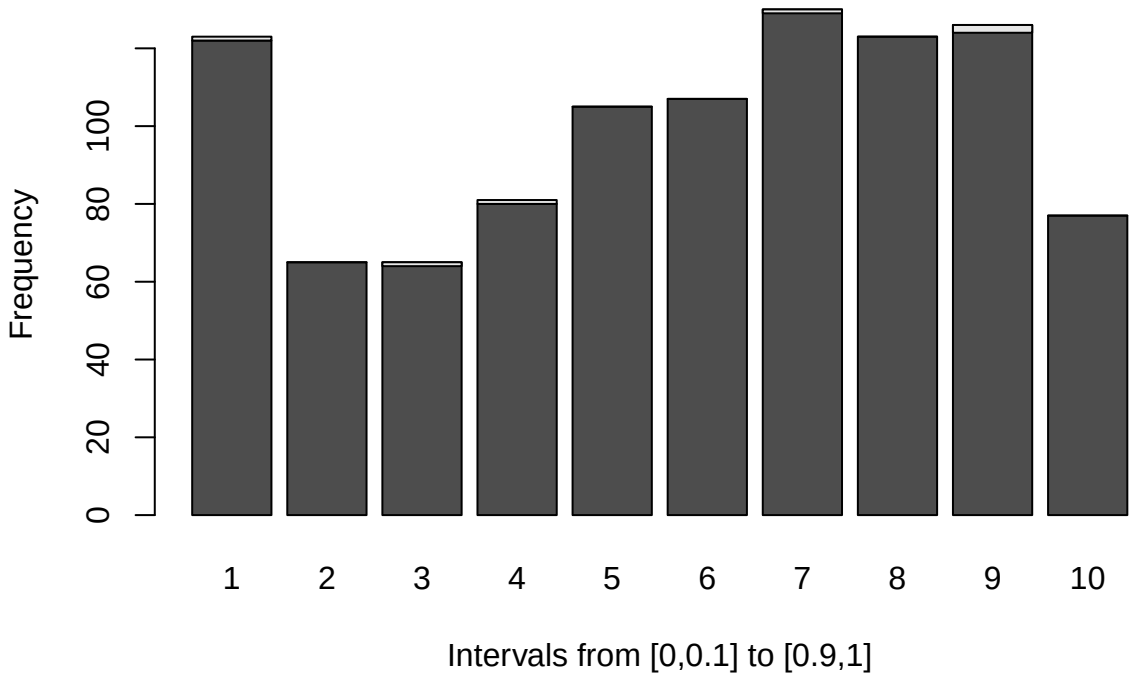
```
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.s.1+x.s.2,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################
PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
lowerl=PI[,2]
```

```
upperl=PI[,3]

##############################
#Prediction Tolerance Interval
######95% Confidence Intervals
##############################
CI.function(fit)
```




```
## [1] "S is 950; S.star is 951; P.Star is 0.951"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

#########################
#### P Values for KS Test
#########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.0826626437935853; Pvalue.star is 0.00329999999999997"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.089; KS.Pvalue is 0.000726078211583547"
```

```
#This is the actual KS test result
```

4.2.2.1 Different Data Generater & Correct Model
Authentic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1^2 + \hat{\beta}_2 x_2^2 + \hat{\sigma}$

```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
```

```r
X = rnorm.10(n,p) # predictors
X.S = X^2
Y = 1+ X.S[,1] + X.S[,2] +rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Authentic.Data)
```

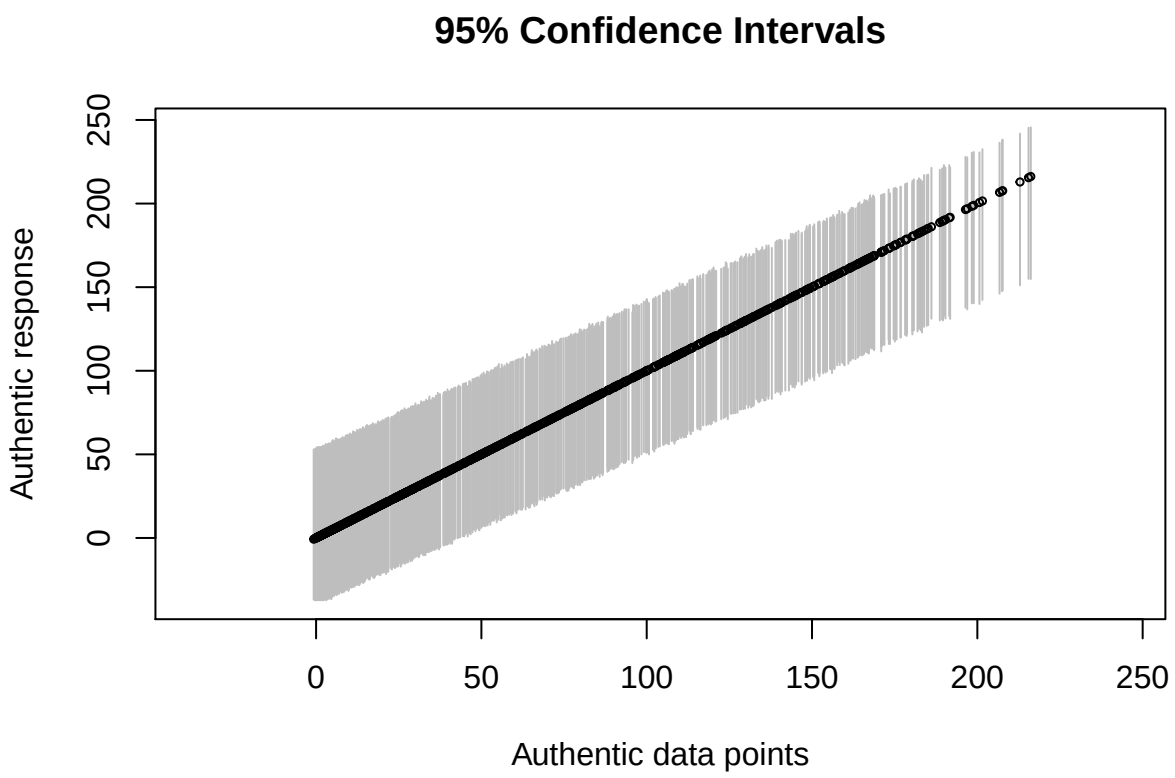
```
##             y         x.1         x.2       x.s.1        x.s.2
## 1  0.33324690 -0.52041548 -0.10001129 0.270832271 0.0100022587
## 2  1.37302656 -0.69433713  1.21617259 0.482104050 1.4790757651
## 3 -0.05646256  0.03771532  0.02155305 0.001422446 0.0004645342
## 4  4.97155983 -0.26172395  2.08022807 0.068499425 4.3273488049
## 5  2.50119188 -0.53030054  1.28237679 0.281218667 1.6444902417
## 6  3.21767981 -1.62743015 -0.09210244 2.648528905 0.0084828599
```

```r
##########################
# Generate synthetic data
##########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
X.S=X^2
aut.fit = lm(y~1+x.1+x.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X[,1]+aut.cof[3]*X[,2]+rnorm(n,0,aut.sig) #response variable
Synthetic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Synthetic.Data)
```

```
##            y        x.1         x.2      x.s.1       x.s.2
## 1  10.490026  1.5719486 -0.44467412 2.47102243 0.197735072
## 2 -33.959492  1.1823048 -0.05650332 1.39784471 0.003192626
## 3  -2.774593 -1.0179358  0.83385415 1.03619329 0.695312737
## 4  -8.778957  1.9116967  0.32589183 3.65458439 0.106205486
## 5 -23.970174  0.1341788 -0.56915998 0.01800395 0.323943082
## 6   2.858564  0.9557308  0.55204255 0.91342145 0.304750975
```

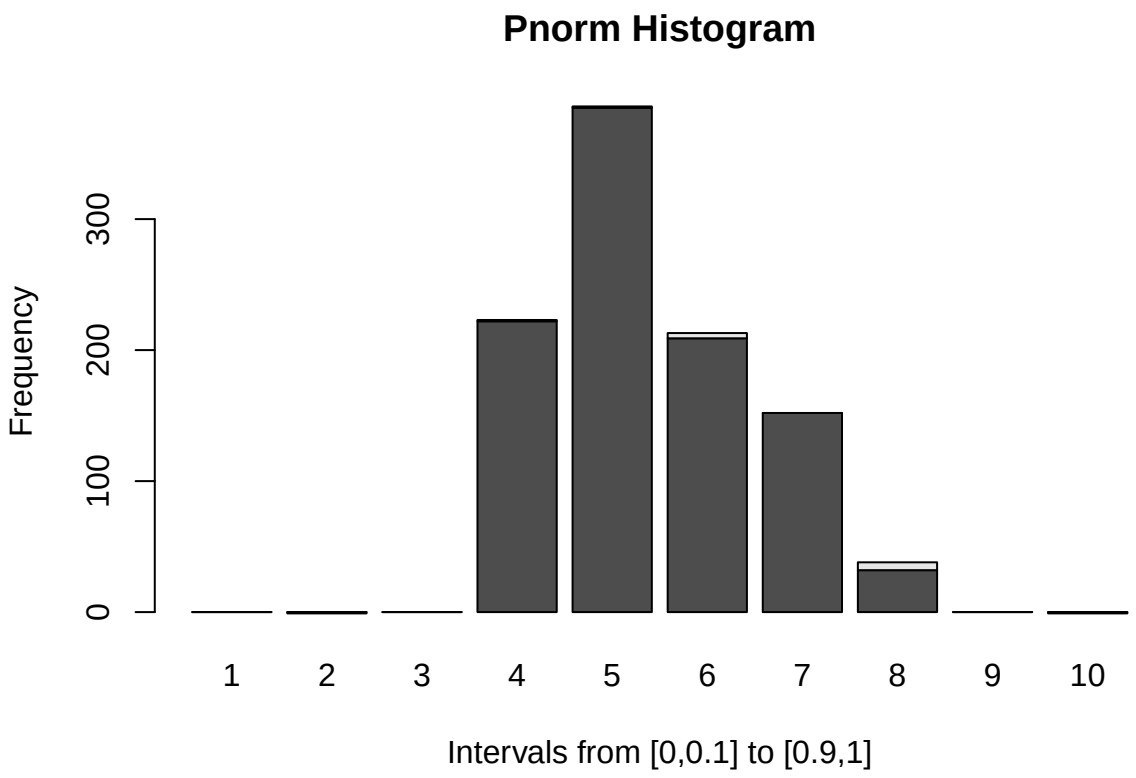
```r
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.s.1+x.s.2,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################
PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
lowerl=PI[,2]
upperl=PI[,3]
```

```
##############################
#Prediction Tolerance Interval
######95% Confidence Intervals
##############################
CI.function(fit)
```




```
## [1] "S is 1000; S.star is 1000; P.Star is 1"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

#########################
#### P Values for KS Test
#########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.103672679755792; Pvalue.star is 9.9999999999989e-05"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.121; KS.Pvalue is 8.76041108655023e-07"
```

```
#This is the actual KS test result
```

4.2.2.2 Different Data Generater & Correct Model
Authentic database generator: $y = \alpha + \beta_1 x_1^2 + \beta_2 x_2^2 + \sigma$
Synthetic database generator: $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \sigma$
Prediction Model: $\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\sigma}$

```
#########################
# Generate authentic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
sigma = 1 # Standard deviation for the error
```

```
X = rnorm.10(n,p) # predictors
X.S = X^2
Y = 1+ X.S[,1] + X.S[,2] +rnorm(n,0,sigma)  # response variable

Authentic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Authentic.Data)
```

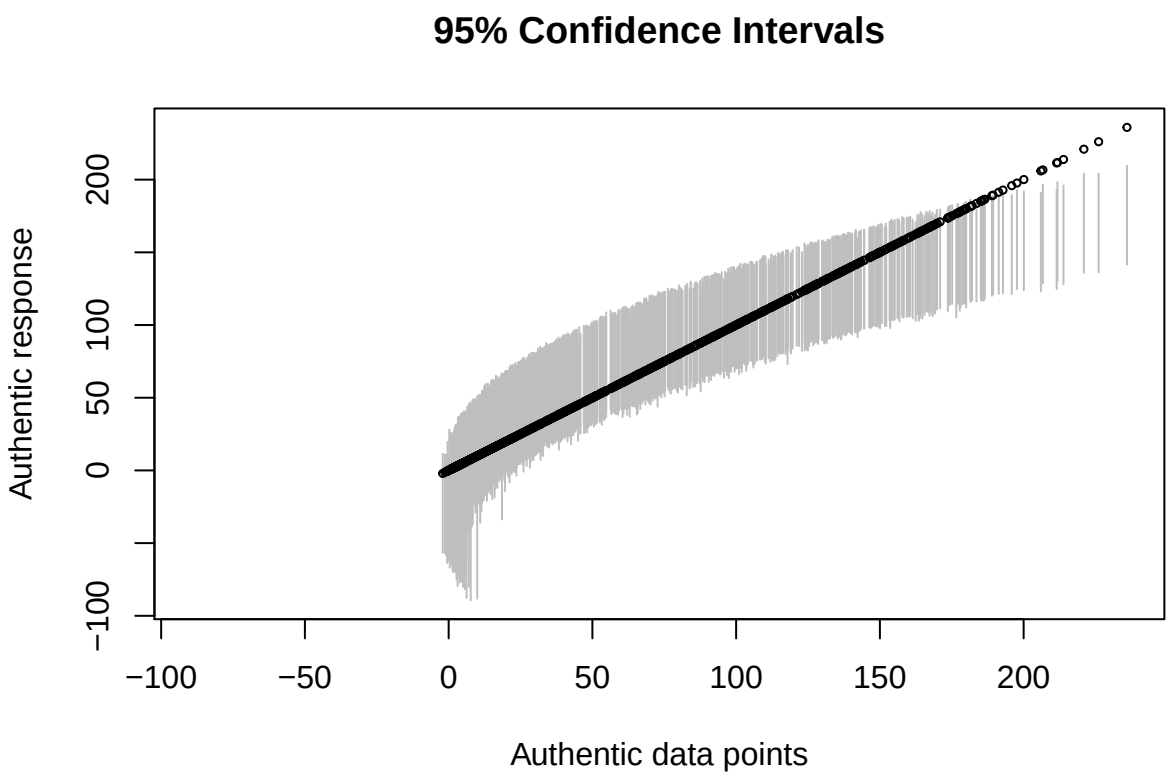
```
##          y        x.1        x.2      x.s.1     x.s.2
## 1 4.631297 -0.7840594 -1.4847968 0.61474909 2.2046215
## 2 4.077442  1.5437903  0.4386689 2.38328842 0.1924304
## 3 1.149227  0.1164936  0.6000420 0.01357076 0.3600504
## 4 4.365599  1.7822542  0.5729597 3.17642987 0.3282828
## 5 6.213115  1.1271405 -2.0434577 1.27044572 4.1757195
## 6 4.912094  1.8966002  0.4569455 3.59709231 0.2087992
```

```
#########################
# Generate synthetic data
#########################
p = 2 # number of predictor
n = 1000 # sample size
B = rep(1,p+1) # regression coefficients
sigma = 1 # Standard deviation for the error
X = rnorm.10(n,p) # predictors
X.S=X^2
aut.fit = lm(y~1+x.1+x.2,data=Authentic.Data)
aut.cof = as.vector(aut.fit$coefficients)
aut.sig = summary(aut.fit)$sigma
Y = aut.cof[1]+ aut.cof[2]*X[,1]+aut.cof[3]*X[,2]+rnorm(n,0,aut.sig) #response variable
Synthetic.Data = data.frame(y=as.vector(Y),x=X,x.s=X.S)
head(Synthetic.Data)
```

```
##            y        x.1        x.2      x.s.1     x.s.2
## 1 -10.719160  0.2103758  0.8551226 0.04425799 0.7312346
## 2 -54.504769 -0.8803121 -0.4532179 0.77494946 0.2054064
## 3   8.534683  0.6878618 -0.4194636 0.47315380 0.1759497
## 4 -50.751657 -1.7728803 -0.6614119 3.14310453 0.4374658
## 5 -34.157069 -2.0032586  1.4987939 4.01304518 2.2463832
## 6 -19.646666 -0.1932214  0.7312798 0.03733450 0.5347701
```

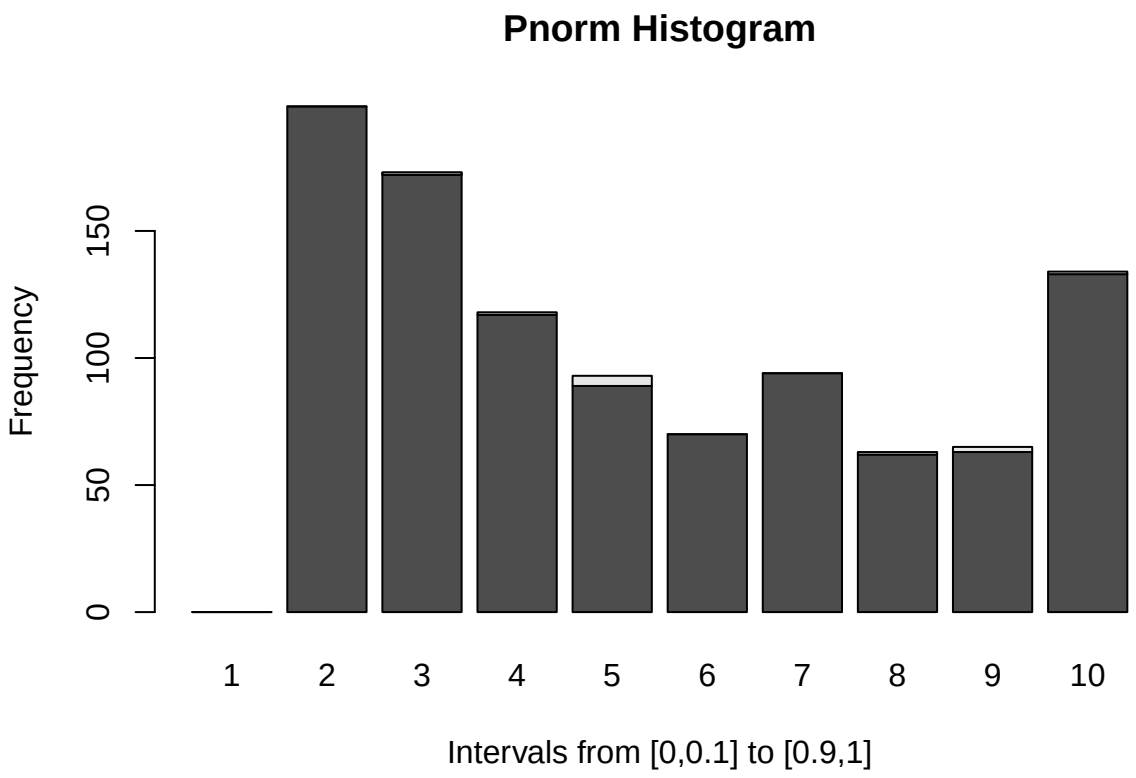
```
###############################
# Fit model to synthetic data
###############################
fit = lm(y~1+x.1+x.2,data=Synthetic.Data)

###################################
# Compute the Prediction Mean Value
# Compute the lower and upper bound
###################################
PI=predict(fit,newdata = Authentic.Data,interval='prediction')
Mean=PI[,1]
lowerl=PI[,2]
upperl=PI[,3]
```

```
##############################
#Prediction Tolerance Interval
######95% Confidence Intervals
##############################
CI.function(fit)
```




```
## [1] "S is 952; S.star is 954; P.Star is 0.954"
```

```
##############################
# Draw the HIstogram of P-norm
##############################
Pnorm.function(fit)
```




```
# This is the Pnorm Histogram to release in the second algorithm

#########################
#### P Values for KS Test
#########################
ks.function(Mean,fit,Authentic.Data$y)
```

```
## [1] "Differentially Private: KS.Star is 0.155795307729877; Pvalue.star is 0"
```

```
#This is the ks test result to release in the third algorithm
ks.actual(Mean,fit,Authentic.Data$y)
```

```
## [1] "Actual KS test KS.statistics is 0.165; KS.Pvalue is 3.00170999167904e-12"
```

```
#This is the actual KS test result
```