MATH 535 HOMEWORK 3

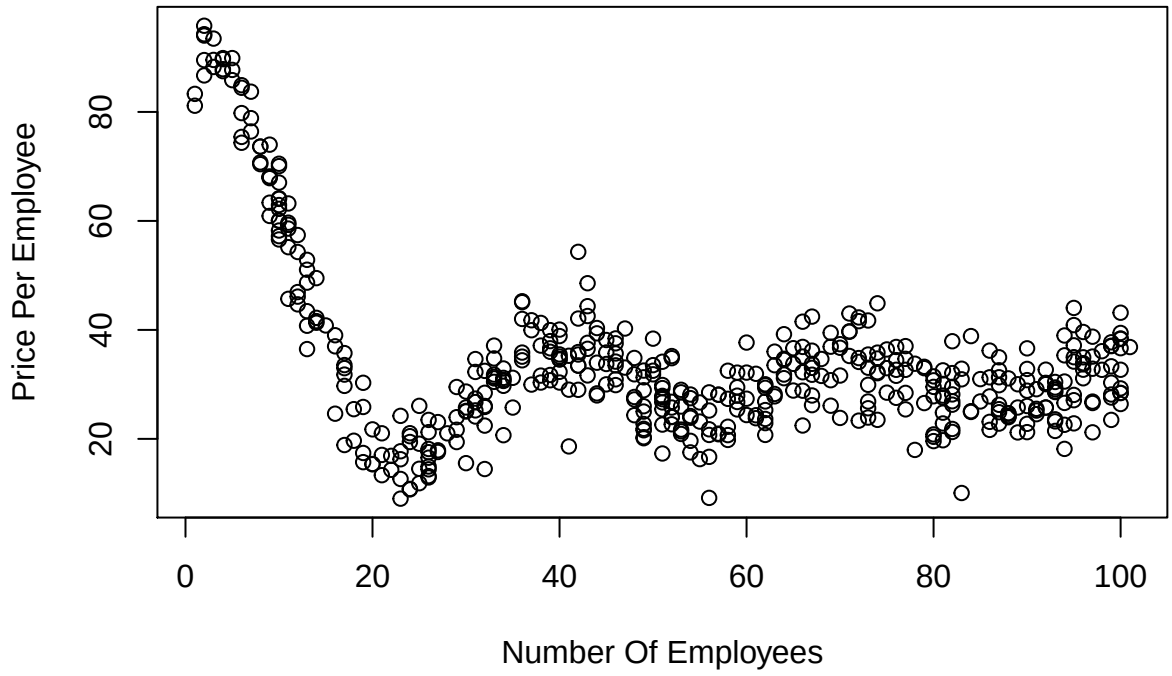
Gustavo Esparza

3/28/2019

```
library(readr)
CostData = read_csv("/Users/gustavo/Desktop/Math 535/8/Cost per employee.csv")

y=CostData$cost
x=CostData$employee
n=length(x)
```

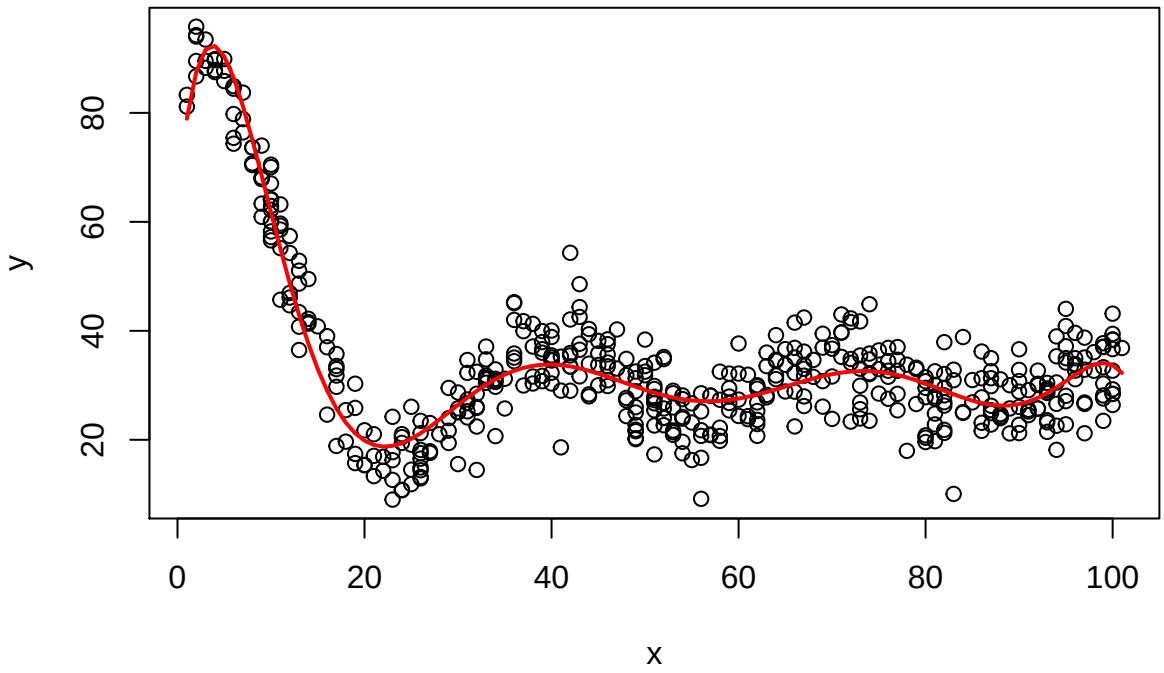
1.) We will begin by plotting our data and determining if a parametric model would be appropriate.

```
plot(x,y, xlab="Number Of Employees", ylab="Price Per Employee")
```



The trend of the data appears to be quite complicated. Using a parametric model, we will attempt to find an appropriate model of some power, n and determine if the model has healthy residuals. Since a simple linear model is obviously not a good model for this particular data, a trial of various powers of x was tested for minimized AIC. For example, the following parametric model was found to fit the data:

```
model1 = lm(y~x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8))
plot(x,y)
x.seq = seq(min(x),max(x),by=1)
y.seq1 = predict(model1,newdata=data.frame(x=x.seq))
plot(x,y)
lines(x.seq,y.seq1,col=2,lwd=2)
```

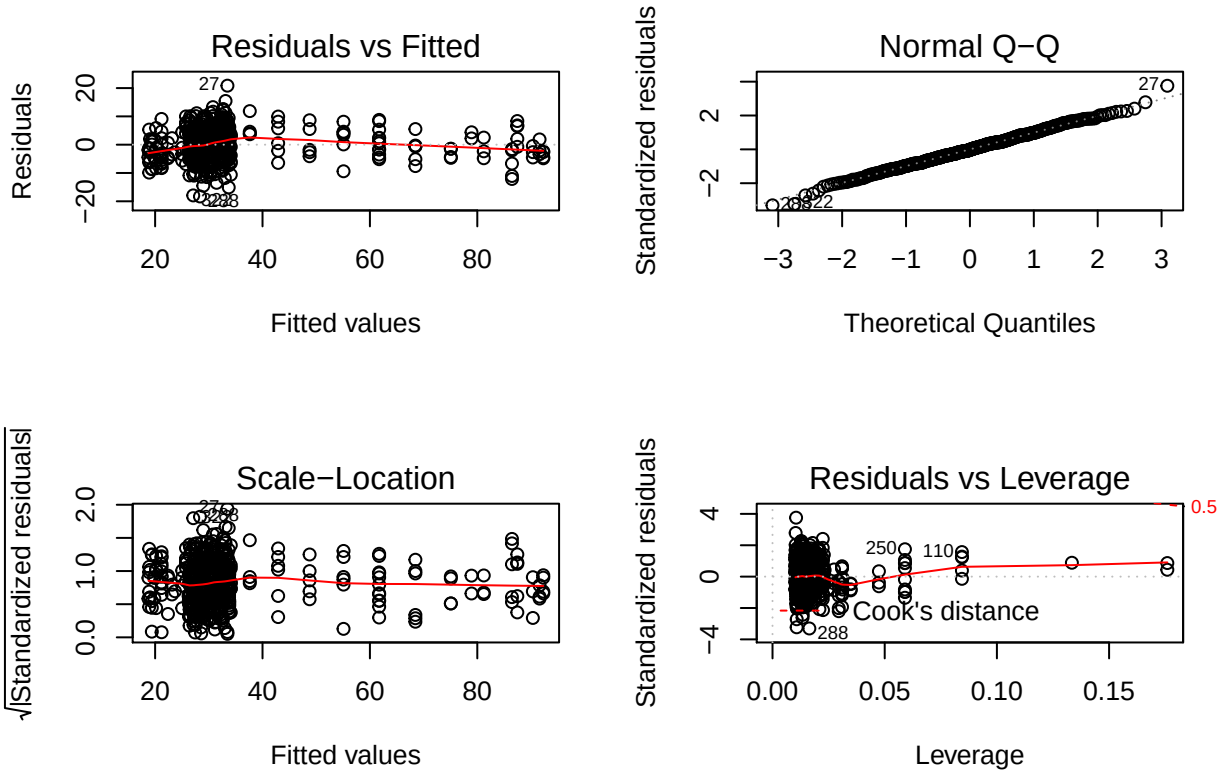



```
AIC(model1)
```

```
## [1] 3149.115
```

The model does appear to go through a good amount of our data and have a smaller AIC than previous power attempts, which is a good sign. However, we must still analyze our residual plots to see if our assumptions are met for this model.

```
par(mfrow=c(2,2))
plot(model1)
```



Clearly, we can see that the residuals vs Fitted plot and the Scale-Location plot do not have residuals randomly scattered across the X-axis. Thus, all of our assumptions are not met for a parametric regression model. Rather than attempt to correct the violations of our residuals, we will instead focus on a non-parametric model to fit our data.For this particular set of data, we will utilize a

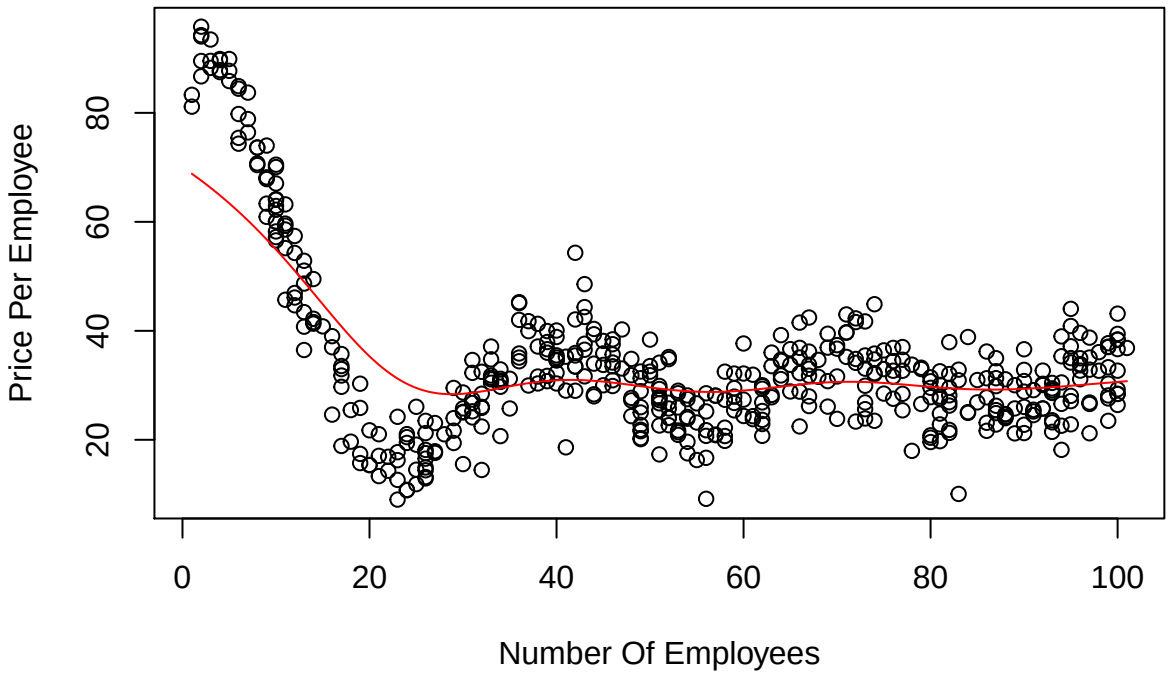
Gaussian Kernel Regression model.

We will first attempt our Kernel regression by using a bandwidth, h, determined by Silverman's Rule of Thumb.

```
h = 1.06*sd(x)*length(x)^(-.2)

x1 = seq(min(x),max(x),by=(max(x)-min(x))/(n-1))
y1 = rep(0,n)

for(i in 1:n){
  y1[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-x)/h)^2)*y)/
          sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-x)/h)^2))}
```

Now, we will plot our data along with our Kernel Regression to determine how well we fit our data.

```
plot(x,y, xlab="Number Of Employees", ylab="Price Per Employee")
lines(x1,y1,col="red")
```



It appears that our Kernel regression using Silverman's rule fit our data quite poorly. Our data has many peaks and valleys, so we should attempt to fit a more robust kernel regression. We will attempt to improve our model by finding a more suitable bandwidth, h.

We will find our new h by minimizing the SSR and also utilizing ten-fold cross validation.

```
index = sample(1:length(x),length(x),replace=F)
n.test = floor(length(x)/10)

CVSSR = function(h){
  res.cv = rep(0,length(x))
  for(j in 1:10){
    test.x = x[index[(1 + (j-1)*50):(j*50)]]
    test.y = y[index[(1 + (j-1)*50):(j*50)]]
    train.x = x[-index[(1 + (j-1)*50):(j*50)]]
    train.y = y[-index[(1 + (j-1)*50):(j*50)]]
    x1 = test.x
    y1 = rep(0,n.test)
    for(i in 1:n.test){
    y1[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-train.x)/h)^2)*train.y)/
            sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-train.x)/h)^2))}
    res.cv[(1 + (j-1)*50):(j*50)] = test.y - y1
  }
  SSR = sum(res.cv^2)
  SSR
}

h2=optim(h,CVSSR)$par
```

Thus, our new bandwidth is a result of the previous optimization:

```
h2
```

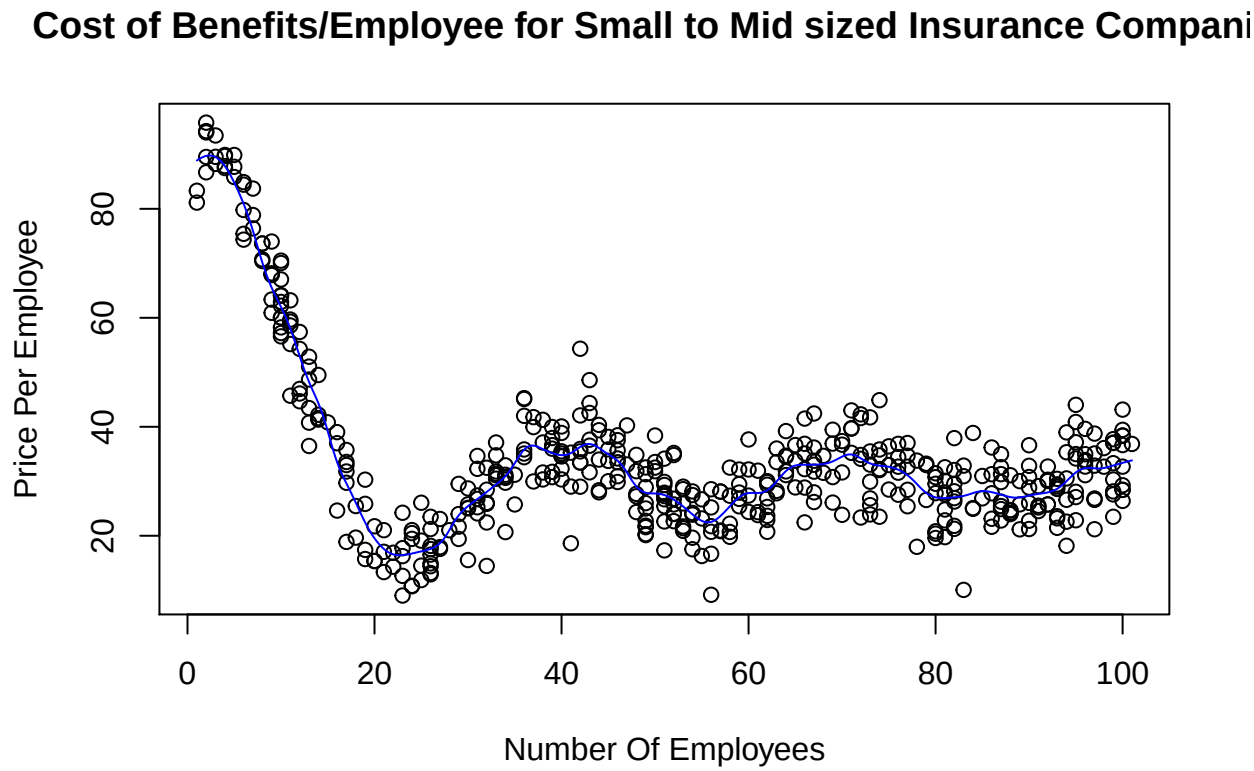
```
## [1] 1.212356
```

An important note regarding our cross validated optimization is the fact that our new bandwidth will differ depending on which values were left out during the cross validation. For the sake of consistency, we will use an h value of 1.31.

```
h2=1.31
```

```
x2 = seq(min(x),max(x),by=(max(x)-min(x))/(n-1) )
y2 = rep(0,n)
for(i in 1:n){
  y2[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x2[i]-x)/h2)^2)*y)/
          sum((1/(sqrt(2*pi)))*exp(-.5*((x2[i]-x)/h2)^2))}
```

```
plot(x,y, xlab="Number Of Employees", ylab="Price Per Employee",
     main="Cost of Benefits/Employee for Small to Mid sized Insurance Companies")
lines(x2,y2,col="blue")
```



Our new bandwidth gives us a model that fits the data much better than using Silverman's rule of thumb. We can see that our Kernel model goes through a majority of our data, and also manages to adjust to any shifts in the general trend of the data.

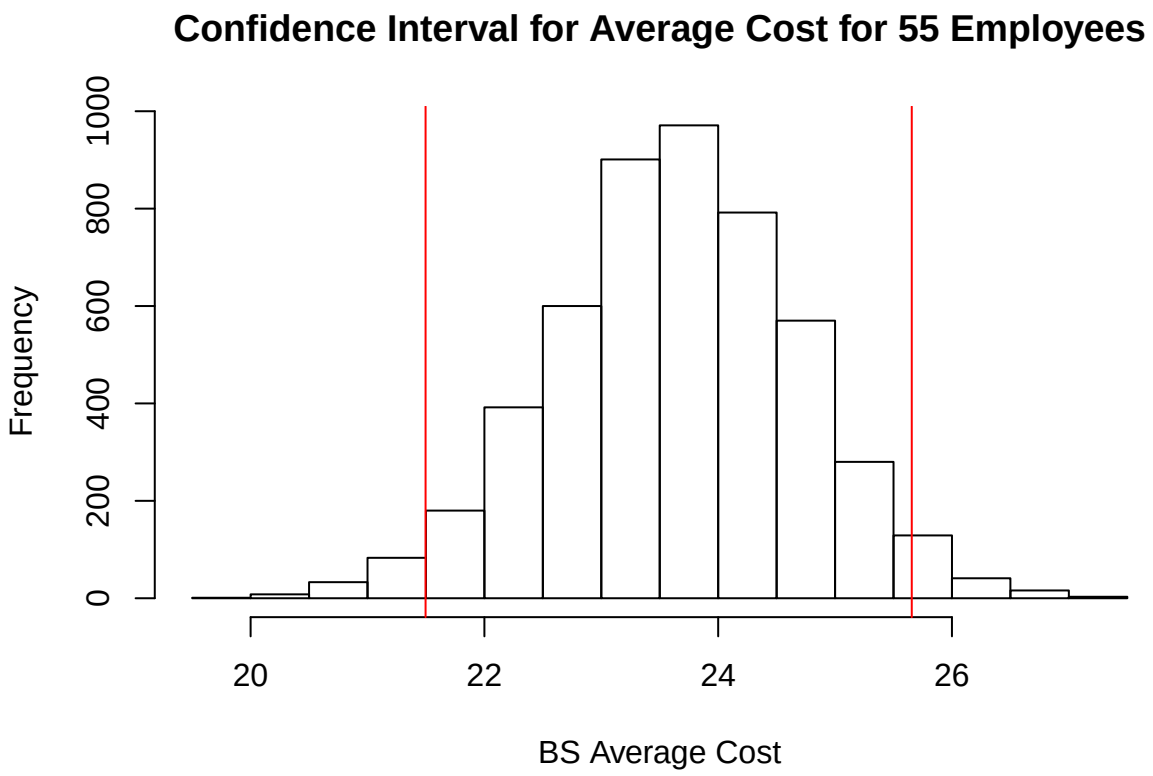
2.) In order to construct a Confidence Interval for our Kernel Regression model, we must first create our residuals.

```
yhat =rep(0,n)

for(i in 1:n){
  yhat[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x[i]-x)/h2)^2)*y)/
            sum((1/(sqrt(2*pi)))*exp(-.5*((x[i]-x)/h2)^2))}

residuals = y-yhat
```

Now, we will construct our confidence interval by bootstrapping new fitted values and, using our residuals, new y values. Then, we will bootstrap mean values when x (the number of employees) is equal to 55.

```
B.NPBS = rep(0,5000)

for(i in 1:5000){
  #x values
  new.x = x[sample(1:length(x),length(x),replace=T)]


  fit.y=rep(0,n)
  new.y=rep(0,n)

  #fitted values
  for(j in 1:n){
    fit.y[j] = sum((1/(sqrt(2*pi)))*exp(-.5*((new.x[j]-x)/h2)^2)*y)/
               sum((1/(sqrt(2*pi)))*exp(-.5*((new.x[j]-x)/h2)^2))}

  #new values with added residuals
   new.y = fit.y + sample(residuals,n,replace=T)

  #bootstrapped values at our desired number of employees
  B.NPBS[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((new.x-55)/h2)^2)*new.y)/
              sum((1/(sqrt(2*pi)))*exp(-.5*((new.x-55)/h2)^2))}
```

```
l=sort(B.NPBS)[125]
u=sort(B.NPBS)[4875]

hist(B.NPBS, xlab="BS Average Cost", ylab="Frequency",
     main="Confidence Interval for Average Cost for 55 Employees")
abline(v=l, col="red")
abline(v=u, col="red")
```



We can see that our bootstrapped values have a normal distribution, which is what we expected. Using symmetry, we are able to declare the 95% interval bounds. Doing so, we have the following 95% Confidence Interval for average cost in benefits for companies that have 55 employees.

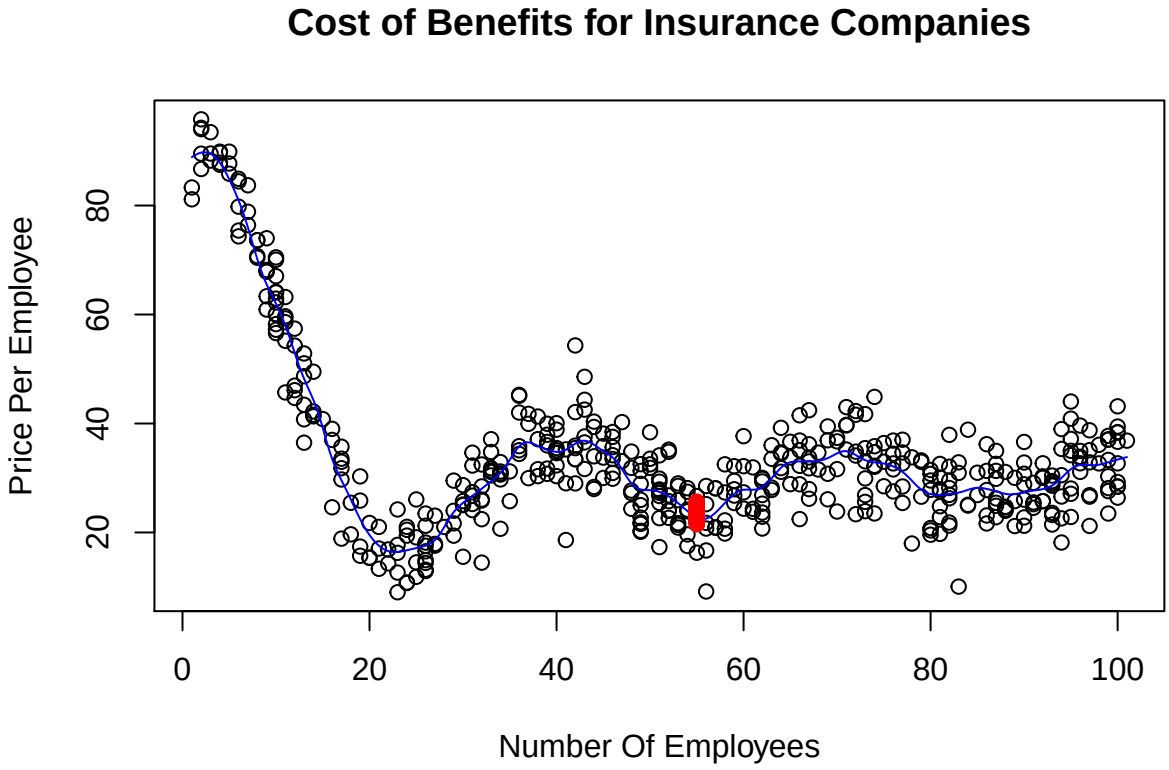
```
cat("\nConfidence Interval = [",l,",", u,"]")
```

```
##
## Confidence Interval = [ 21.49682 , 25.65622 ]
```

Thus, we can say that we are 95% confident that the true average cost in benefits for a company that has 55 employees is between these two values.

We can also illustrate our confidence interval using our original plot of the data.

```
plot(x,y, xlab="Number Of Employees", ylab="Price Per Employee",
     main="Cost of Benefits for Insurance Companies")
lines(x2,y2,col="blue")
ci_avg = seq(l, u, length.out = 50)
ci_x = rep(55, 50)
lines(ci_x, ci_avg, col = "red", type = "p")
```

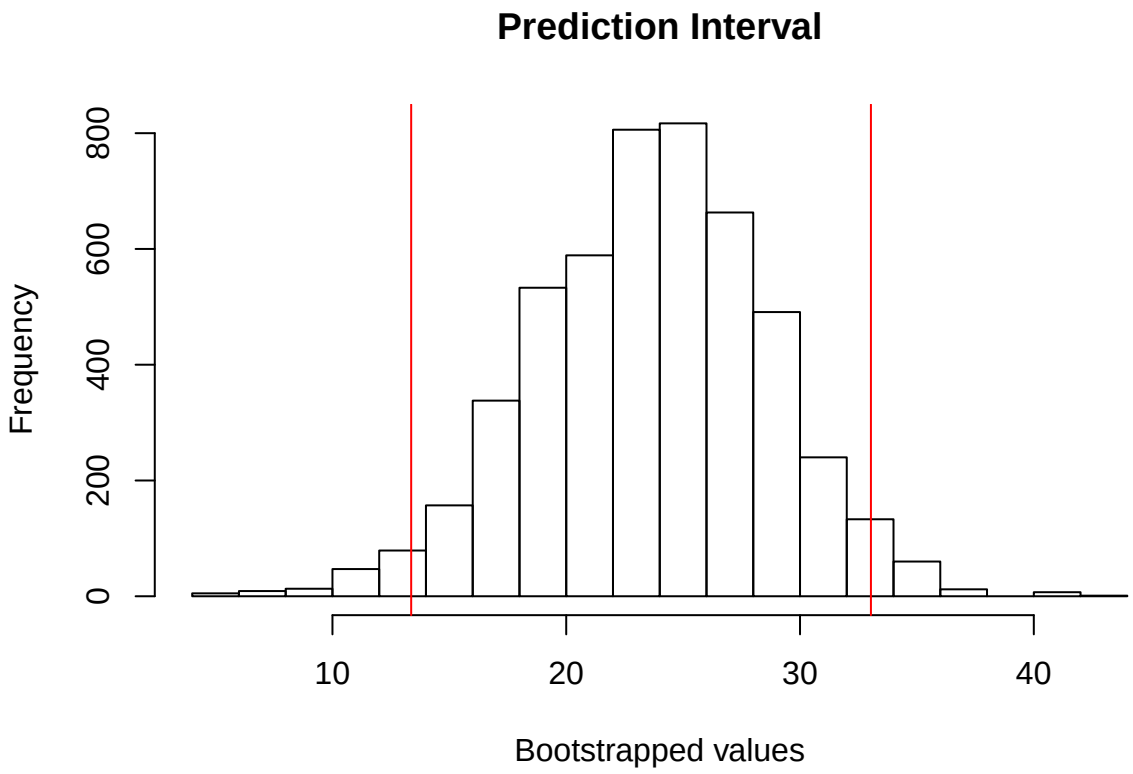


We have plotted our data, kernel regression curve and the confidence interval (the red vertical line). Thus, when there are 55 employees in a company, we can observe that our confidence interval contains our kernel regression curve. This is a good indication that our bootstrapped confidence interval was succesful in determining an interval for the average cost in benefits at this value.

C

Now, for our prediction interval we will add residuals to the bootstrapped values found in our confidence interval in part **B**. We will then display our histogram for the bootstrapped Prediction Interval values.

```
B.NPBS2 = rep(0,5000)
B.NPBS2 = B.NPBS + sample(residuals,5000,replace=T)

lp=sort(B.NPBS2)[125]
up=sort(B.NPBS2)[4875]

hist(B.NPBS2, xlab="Bootstrapped values", ylab="Frequency",
     main="Prediction Interval", breaks=25)
abline(v=lp, col="red")
abline(v=up, col="red")
```



Again, our bootstrapped values have a normal bellcurve shape. We can observe that our Prediction interval is wider than the Confidence Interval previously computed. This makes sense, since we are no longer trying to place bounds on the average cost but rather attempting to construct an interval for a single cost for a company with 55 employees. By definiton, this single value should have more variability than a computed average for multiple companies.

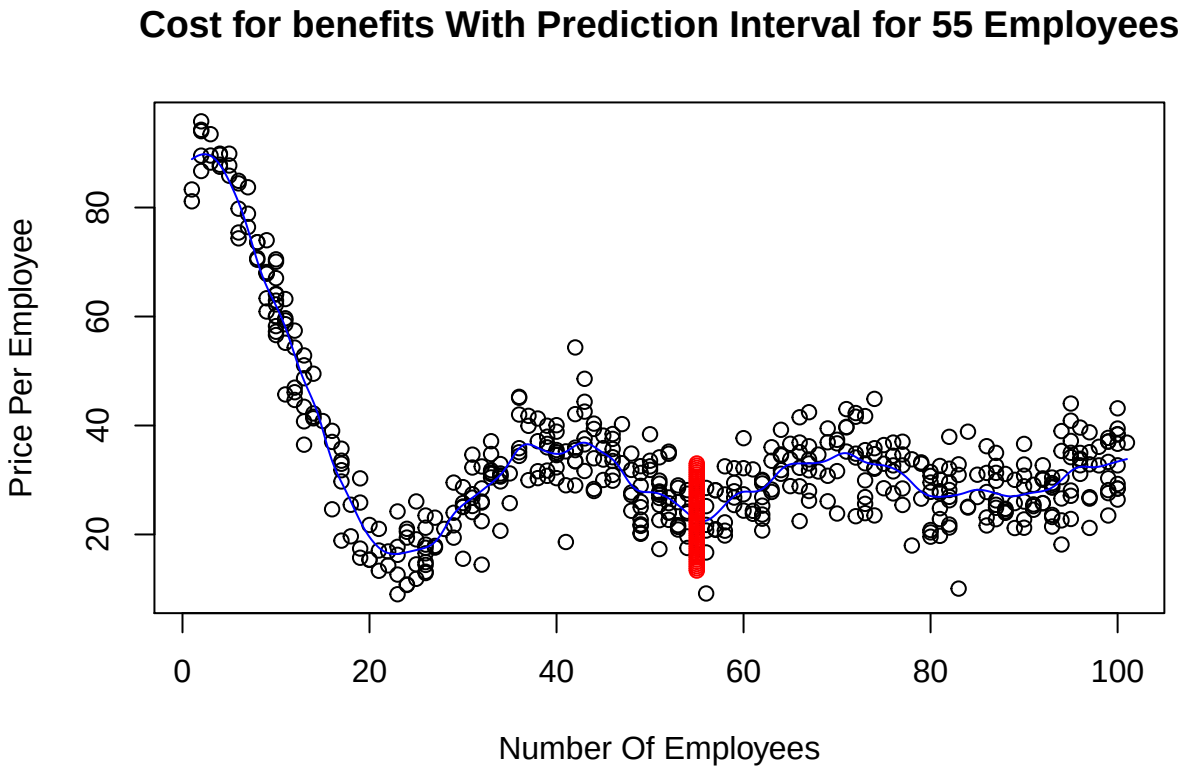
Finally we can provide a straightforward prediction interval as follows:

```
cat("\nPrediction Interval = [",lp,",", up,"]")
```

```
##
## Prediction Interval = [ 13.3681 , 33.033 ]
```

Thus, we are 95% confident that the average cost for a single company that has 55 employees is between these two values.

```
plot(x,y, xlab="Number Of Employees", ylab="Price Per Employee",
     main="Cost for benefits With Prediction Interval for 55 Employees")
lines(x2,y2,col="blue")
pi_avg = seq(lp,up, length.out = 50)
pi_x = rep(55, 50)
lines(pi_x, pi_avg, col = "red", type = "p")
```



Again, we can display our prediction interval using our plotted data and kernel regression model.We can still see that the prediction interval is much wider than our confidence interval, but stil seems to accurately predict the range of response values for our specific x value of interest.