Math 536 Exam 1

Shamir Naran

Due March 15, 2020

```r
data <- read.csv("Health Care Cost per Employee.csv", header=T, na.strings="?")
# We read in the data using the read.csv function.
# The header = T tells R that the first line of the file contains the variable names.
# na. strings tells R that when it sees a particular set of characters,
# it should be treated as a missing element.
```

```r
library(tidyverse) # loads readr, dplyr, ggplot2 and other useful packages
```

1.

In this dataset, you will find data on small to mid-sized local business and their health care costs (in thousands). There are two variables, the first is the number of employees that a company has. This number ranges from a single employee up to about 100 employees. The second variable represents the average cost in benefits associated with employees.

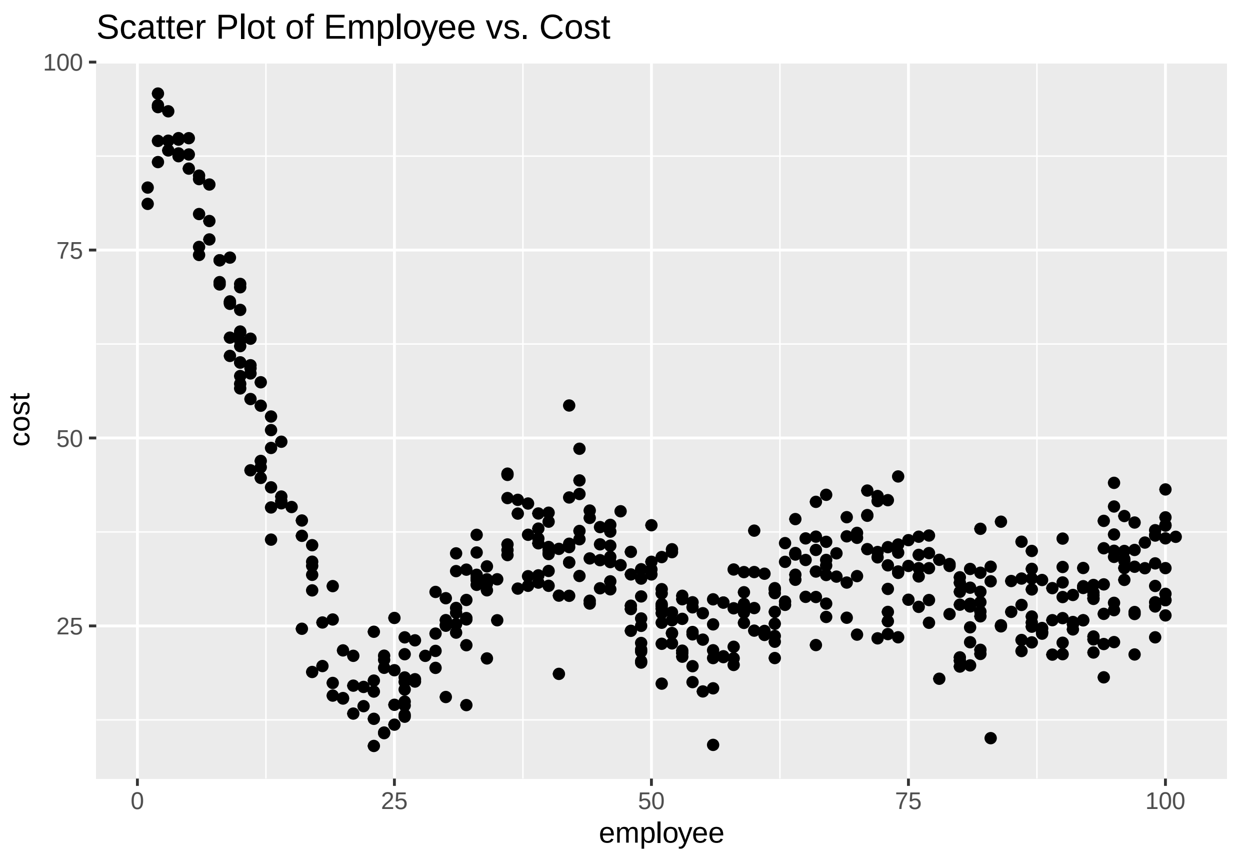
We want to develop a model for estimating the average or expected average cost of benefits based on the number of employees a company has.

When we plot the data, we see that the relationship between employee and cost is non-linear. There isn't a function we can use to estimate the average cost of benefits based on the number of employees. We're best off using a non-parametric method like Kernel Regression.

Kernel regression is a non-parametric technique in statistics to estimate the conditional expectation of a random variable. The idea is to estimate the weighted sum of all observed y values for a given predictor value, x_i. Weights are the kernel values, scaled between 0 and 1. We will use the Gaussian density as our kernel function.

When using the Gaussian density as our kernel function, "h" plays the important role of our bandwidth. This parameter controls the smoothness of your model. Smaller values will cause the model to overfit the data and larger values will cause it to underfit. We want to pick the h that minimizes some objective measure of fit such as sum of squared residuals.

```r
ggplot(data = data, mapping = aes(x = employee, y = cost)) +
  geom_point() +
  labs(title = "Scatter Plot of Employee vs. Cost")
```



We use k-fold cross validation to determine our h value. Our data is partitioned into 90% training and 10% testing. We create a model using our training data and evaluate objective measure of fit using our test data. We rotate the test data k times.

```r
set.seed(99) # for reproducibility
j = 1
k = 10
n = length(data$employee)
samp.index = sample(1:n,n,replace = F)
rows = k
cols = floor(n/k)
samp.matrix = matrix(samp.index[1:(cols*rows)],nrow=rows)
```

```r
# K-Fold Cross-Validation to determine bandwidth (h) for Kernel Regression

bandwidth.sel = function(h){
    RSS = 0
    for(j in 1:k){
        test = data[samp.matrix[j,],]
        train = data[-samp.matrix[j,],]
        x = train$employee
        y = train$cost
        x1 = test$employee
        y.test = test$cost

        y1 = rep(0,length(x1))
        for(i in 1:length(x1)){
            y1[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i]-x)/h)^2)*y)/sum((1/(sqrt(2*pi)))*exp(-.5*((x1[i
        }

        RSS = RSS + sum((y.test - y1)^2)
        #Note, I'm recursively adding to RSS, which I set to 0 outside my loop
    }
    RSS
}
```

```
# use the optim function to minimze h

h = optim(10,bandwidth.sel)$par
```

```
## Warning in optim(10, bandwidth.sel): one-dimensional optimization by Nelder-Mead is unreliable:
## use "Brent" or optimize() directly
```

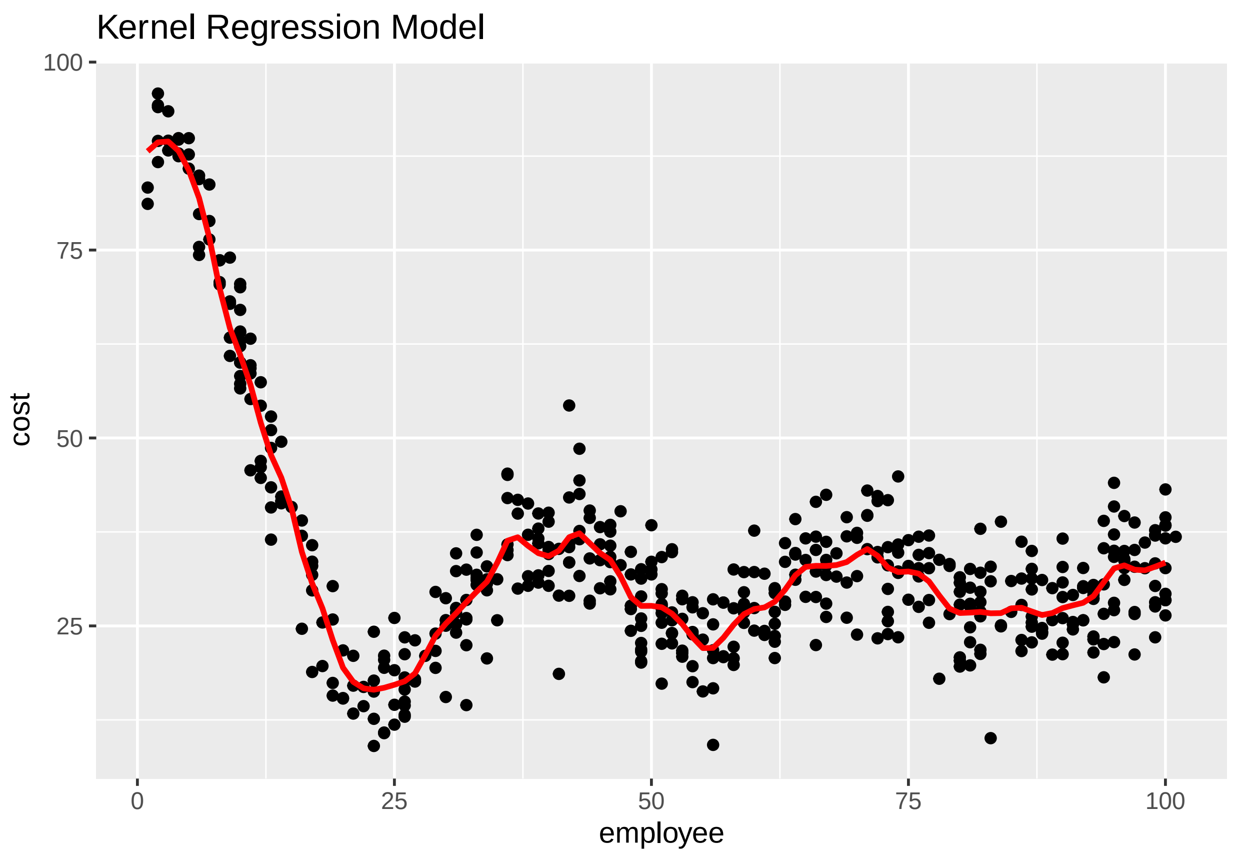
```
h
```

```
## [1] 1.231445
```

We get an h value of 1.231445. Now let's use that h in our kernel density and fit the model.

```
test = data[samp.matrix[j,],]
train = data[-samp.matrix[j,],]
# x and y are the data we are storing to make the model
x = train$employee
y = train$cost
# x1 is the new input values that which I want to make a prediction
# y1 is the new predicted values
x.model = seq(1,100,by = 1)
y.model = rep(0,100)
for(i in 1:100){
    y.model[i] = sum((1/(sqrt(2*pi)))*exp(-.5*((x.model[i]-x)/h)^2)*y)/sum((1/(sqrt(2*pi)))*exp(-.5*((x
}

ggplot(data = data, mapping = aes(x = employee, y = cost)) +
  geom_point() +
  geom_line(data = data.frame(x.model,y.model),
            mapping = aes(x = x.model, y = y.model), col = "red", size = 1) +
  labs(title = "Kernel Regression Model")
```



2. & 3.

We are interested in creating a 95% confidence interval for the average cost of benefits per employee for all companies that have 55 employees and a 95% prediction interval. To do this we need to bootstrap. Let's check our assumption of constant variance.

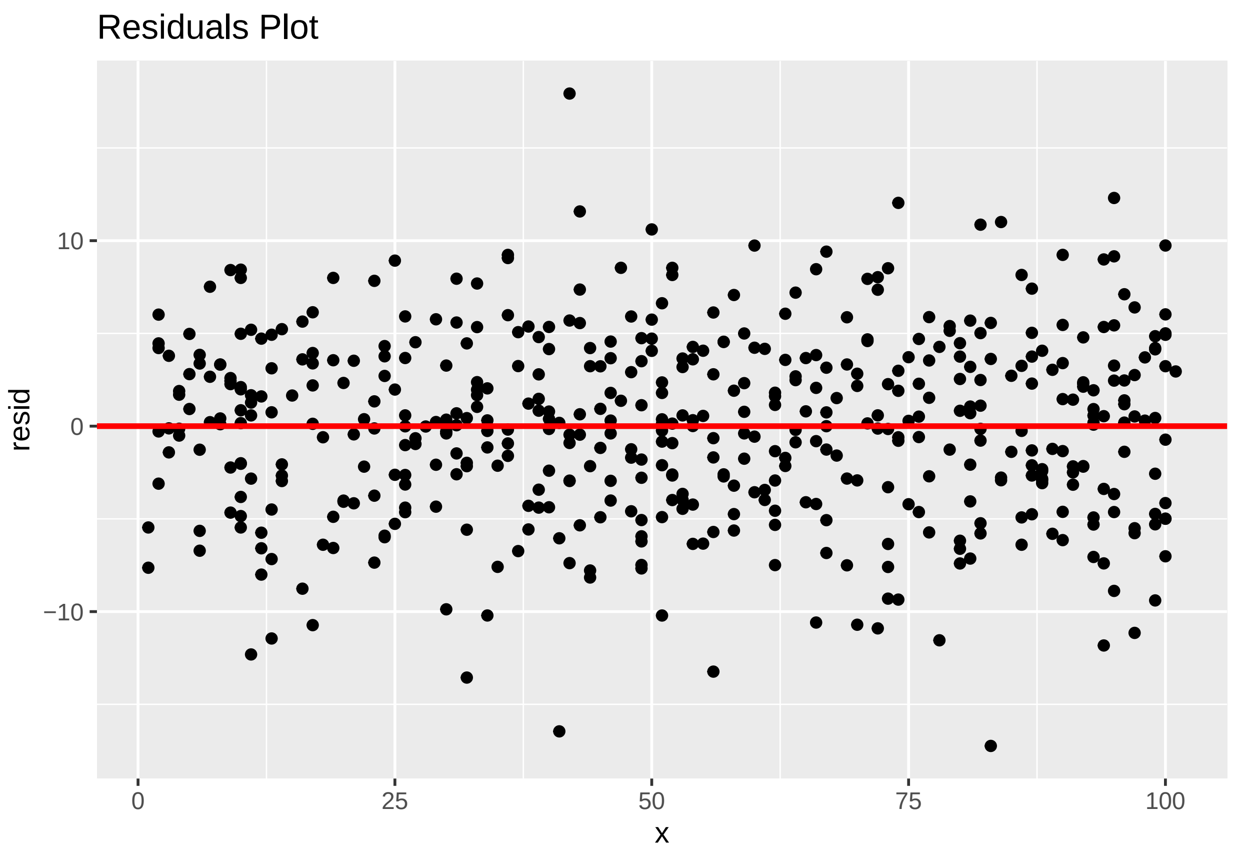
```
x = data$employee
y = data$cost

x1 = x # I want the input to be the original x values
y1 = rep(0,length(x1)) # initialize y1

for(i in 1:length(x1)){
    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)
}

resid = y - y1 # compute residuals

ggplot(data = data, mapping = aes(x = x, y = resid)) +
  geom_point() +
  geom_abline(slope = 0,intercept = 0, col = "red", size = 1) +
  labs(title = "Residuals Plot")
```



My residuals are centered around 0 and we have constant variance. I now have everything I need to bootstrap a confidence interval and a prediction interval.

```
# Bootstrap
# Compute BS.x and BS.y.hat

EY.55 <- rep(0,2000)
PY.55 <- rep(0,2000)

for (i in 1:2000) {

  # I'm still using my original model to compute BS.y.hat
  x = data$employee
  y = data$cost

  BS.x = sample(x,length(x), replace = T)
  # resample the x's with replacement

  x1 = BS.x # input variable should be BS.x
  y1 = rep(0,length(x1)) # initialize y1

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

  # BS.y.hat should be the predicted values at the BS.x location
  BS.y.hat = y1
  # BS. y = BS.y.hat + resampling my original residuals
  BS.y = BS.y.hat + sample(resid, length(resid), replace = T)

# I have a new dataset where I have a BS.x and a BS.y
# I want to use this new dataset to predict y when x is 55
# I need a Kernel Regression model for this too

  x <- BS.x
  y <- BS.y

  x1 = 55 # we want to make the prediction at 55
  y1 = rep(0,length(x1))

  y1 = sum((1/(sqrt(2*pi)))*exp(-.5*((x1-x)/h)^2)*y)/sum((1/(sqrt(2*pi)))*exp(-.5*((x1-x)/h)^2))

  EY.55[i] = y1 # for my confidence interval
  PY.55[i] = EY.55[i] + sample(resid,1) # for my prediction interval
}
```

```
CI <- quantile(EY.55,c(.025,.975)) # 95% confidence interval
PI <- quantile(PY.55,c(.025,.975)) # 95% prediction interval
CI
```

```
##     2.5%    97.5%
## 21.43484 25.63691
```

```
PI
```

```
##     2.5%    97.5%
## 13.05370 32.49077
```

```
CI.lb <- sort(EY.55)[50]
CI.ub <- sort(EY.55)[1950]
PI.lb <- sort(PY.55)[50]
PI.ub <- sort(PY.55)[1950]
```

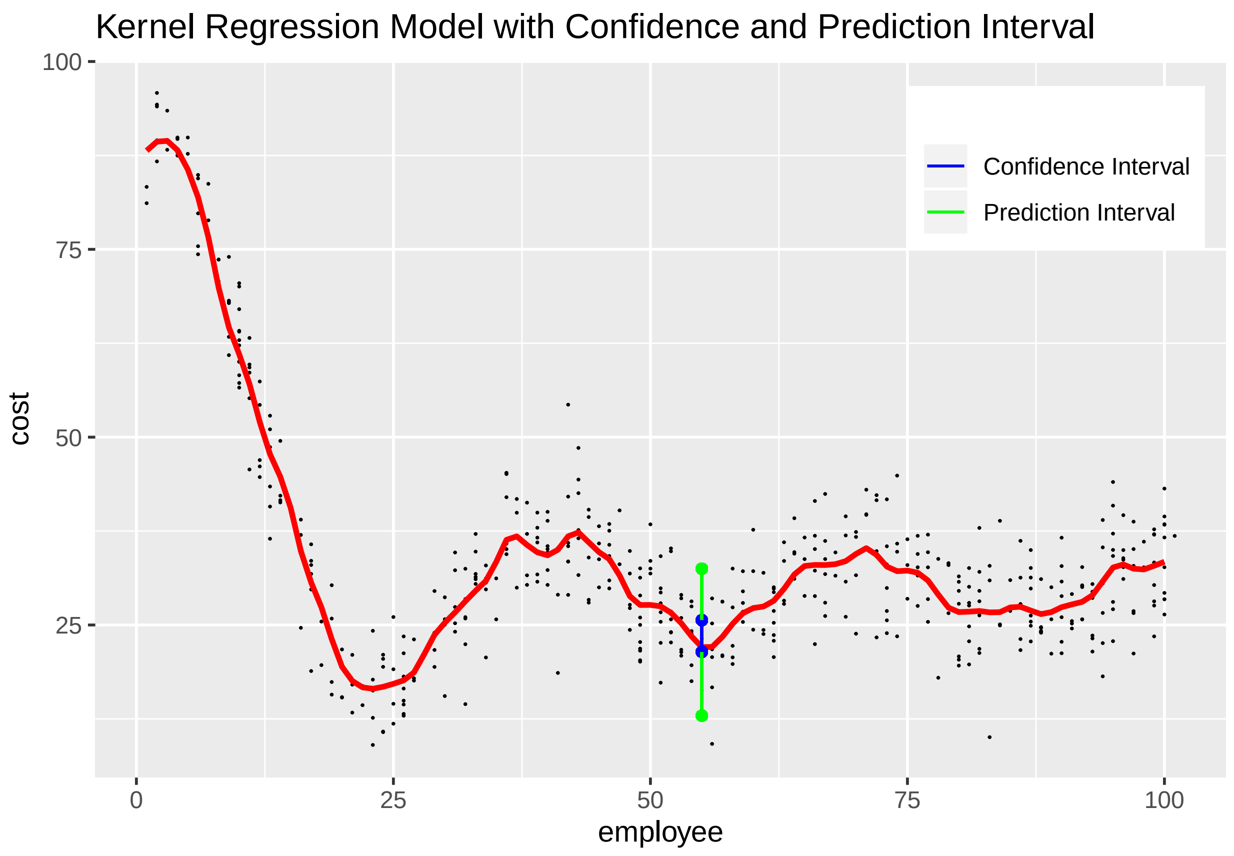
We are 95% confident that the average cost of benefits per employee for all companies that have 55 employees is between 21.35 and 25.51.

We are 95% confident that the average cost of benefits per employee for an individual company that has 55 employees is between 13.03 and 33.52.

4.

We add our intervals from part 2 and 3 to our scatterplot.

```
ggplot(data = data, mapping = aes(x = employee, y = cost)) +
  geom_point(size = 0) +
  geom_line(data = data.frame(x.model,y.model),
            mapping = aes(x = x.model, y = y.model), col = "red", size = 1) +
  geom_point(mapping = aes(x = 55, y = CI.lb), col = "blue") +
  geom_point(mapping = aes(x = 55, y = CI.ub), col = "blue") +
  geom_point(mapping = aes(x = 55, y = PI.lb), col = "green") +
  geom_point(mapping = aes(x = 55, y = PI.ub), col = "green") +
  geom_segment(mapping = aes(x = 55, y = PI.lb, xend = 55, yend = PI.ub, col = "Prediction Interval"))
  geom_segment(mapping = aes(x = 55, y = CI.lb, xend = 55, yend = CI.ub, col = "Confidence Interval"))
  scale_color_manual("",values = c("blue","green")) +
  theme(legend.position = c(0.85,0.85)) +
  labs(title = "Kernel Regression Model with Confidence and Prediction Interval")
```


Kernel Regression Model with Confidence and Prediction Interval
cost
employee
Confidence Interval
Prediction Interval