Assignment 3

Brian Choi

2023-06-12

Question 1

Part 1

```
set.seed(45359407)
N <- 100
X <- rnorm(N, mean = 0, sd = 1)
Y <- X^2
```

Part 2

```
P_hat <- mean(Y <= 1.2)
n_replications <- 500
bootstrap_samples <- replicate(n_replications, mean(sample(Y, replace = TRUE) <= 1.2))
P_hat_star <- mean(bootstrap_samples)
```

Part 3

```
P <- pchisq(1.2, df = 1)
```

Part 4

P_hat is an estimate from the original sample. P_hat_star is an estimate from bootstrapping, which involves resampling from the original sample. The average of the bootstrap estimates gives an estimate of P. P is the exact probability calculated using the cumulative distribution of the chi-square distribution. The discrepancy between P_hat, P_hat_star, and P arises due to sampling variability.

Part 5

```
sample_sizes <- c(100, 250, 500, 1000, 2000, 5000)
n_replications <- 500
proportions <- numeric(length(sample_sizes))
```

```r
for (i in 1:length(sample_sizes)) {
  n <- sample_sizes[i]
  match_count <- 0
  for (j in 1:n_replications) {
    X <- rnorm(n, mean = 0, sd = 1)
    Y <- X^2
    P_hat <- mean(Y <= 1.2)
    bootstrap_samples <- replicate(500, mean(sample(Y, replace = TRUE) <= 1.2))
    P_hat_star<- mean(bootstrap_samples)
    if (abs(P_hat_star -P_hat) < 5e-4) {
      match_count <- match_count + 1
    }
  }
  proportions[i] <- match_count / n_replications
}

for (i in 1:length(sample_sizes)) {
  cat("Sample Size:", sample_sizes[i], "\tProportion:", proportions[i], "\n")
}
```

```
## Sample Size: 100     Proportion: 0.196
## Sample Size: 250     Proportion: 0.316
## Sample Size: 500     Proportion: 0.42
## Sample Size: 1000    Proportion: 0.562
## Sample Size: 2000    Proportion: 0.752
## Sample Size: 5000    Proportion: 0.938
```

Question 2

Part 1

For a function to be a valid kernel function, it needs to have non-negativity and integration equal to 1.

$$c\int_{-1}^{1}(1-x^2)dx = 1$$

$$c[x-\frac{x^2}{3}]_{-1}^{1} = 1$$

$$c[\frac{2}{3}+\frac{2}{3}] = 1$$

$$c = \frac{3}{4}$$

The constant $c = \frac{3}{4}$ makes $K(x)$ a valid kernel function.

Part 2

```r
# Given sample
data <- c(10, 12, 15, 25)
x <- 15
s = sd(data)
n <- length(data)
h <- 1.06*s*n^(-1/5)

# Epanechnikov kernel function
k <- function(t) {
  ifelse(abs(t) <= 1, 3/4 * (1 - t^2), 0)
}

# Kernel density function estimator
f_hat <- function(x) {
  n <- length(data)
  sum_k <- 0

  for (i in 1:n) {
    sum_k <- sum_k + k((x - data[i]) / h)
  }

  (1 / (n * h)) * sum_k
}

# Evaluate the estimator at x = 15
pdf_value <- f_hat(x)

# Print the estimated probability density function at x = 15
cat("The estimated probability density function at x = 15 is:", pdf_value, "\n")
```

```
## The estimated probability density function at x = 15 is: 0.06350468
```

Part 3

```r
# Support of the estimator
support <- c(min(data) - h, max(data) + h)

# Check if the estimator is a valid probability density function
integral <- integrate(f_hat, support[1], support[2])$value

# Print the support and validation results
cat("Support of the estimator:", support[1], "to", support[2], "\n")
```

```
## Support of the estimator: 4.651167 to 30.34883
```

```r
cat("The estimator is a valid probability density function: ", abs(integral - 1) < 1e-6, "\n")
```

```
## The estimator is a valid probability density function:  TRUE
```

```r
# Calculate the expected value of the density function estimator
expected_value <- integrate(function(x) x * f_hat(x), -Inf, Inf)$value

# Calculate the variance of the density function estimator
variance <- integrate(function(x) (x - expected_value)^2 * f_hat(x), -Inf, Inf)$value

# Print the expected value and variance
cat("Expected value of the density function estimator:", expected_value, "\n")
```

```
## Expected value of the density function estimator: 15.49997
```

```r
cat("Variance of the density function estimator:", variance, "\n")
```

```
## Variance of the density function estimator: 38.97194
```

Part 4

Estimating Probability with Kernel Density Estimator

Given a sample of observations: 7, 2, 10, 8, we want to estimate the probability $p = P(\text{A random observation is less than 20})$ using the obtained kernel density function estimator.

Hand Calculation

Estimating Probability with Kernel Density Estimator

Given a sample of observations: 10, 12, 15, 25, we want to estimate the probability $p = P(\text{A random observation is less than 20})$ using the obtained kernel density function estimator.

Hand Calculation

To estimate p, we can use the following hand calculation: - Bandwidth: $h = 1.06sn^{-1/5}$, where s is the sample standard deviation and n is the sample size. - Epanechnikov kernel function: $k(t) = \frac{3}{4}\left(1 - t^2\right)$, $-1 \leq t \leq 1$

$$\bar{X} = \frac{10 + 12 + 15 + 25}{4} = 15.5$$

$$s = \sqrt{\frac{(10 - 15.5)^2 + (12 - 15.5)^2 + (15 - 15.5)^2 + (25 - 15.5)^2}{3}} \approx 6.0415$$

$$h = 1.06sn^{-1/5} = 1.06 \cdot 6.0415 \cdot 4^{-1/5} \approx 2.1837$$

$$p = \int_{-\infty}^{20} \hat{f}(x)dx$$

$$p = \int_{-\infty}^{20} \frac{1}{nh} \sum_{i=1}^{n} k\left(\frac{x - X_i}{h}\right) dx = \frac{3}{4} \cdot \frac{1}{nh} \sum_{i=1}^{n} \int_{-\infty}^{20} \left(1 - \left(\frac{x - X_i}{h}\right)^2\right) dx$$

$$p = \frac{3}{4} \cdot \frac{1}{n \cdot h} \sum_{i=1}^{n} \left[20 - \frac{1}{3}\left(\frac{20 - X_i}{h}\right)^3\right]$$

By calculating the above expression with the given data, $p = 0.75$.

```r
# Estimate p using numerical integration
p_est <- integrate(f_hat, -Inf, 20)$value

# Print the estimated probability p
cat("The estimated probability P(X < 20) is:", p_est, "\n")
```

```
## The estimated probability P(X < 20) is: 0.7500108
```

Part 5

We can use the kernel density estimator and the bootstrap method to find the confidence interval. By generating bootstrapping samples, estimating the kernel density function of each sample, estimating the density function at x = 15, the confidence interval can be obtained from the bootstrapped samples.

Question 3

Part 1

```r
alpha <- 0.4
g1 <- function(x) dchisq(x, df = 3)
g2 <- function(x) dbeta(x, shape1 = 5, shape2 = 2)
g_alpha <- function(x) alpha*g1(x) + (1-alpha)*g2(x)

integrate(g_alpha, 0, Inf)
```

```
## 0.9999999 with absolute error < 8.9e-05
```

Therefore, g_α is a valid probability density function.

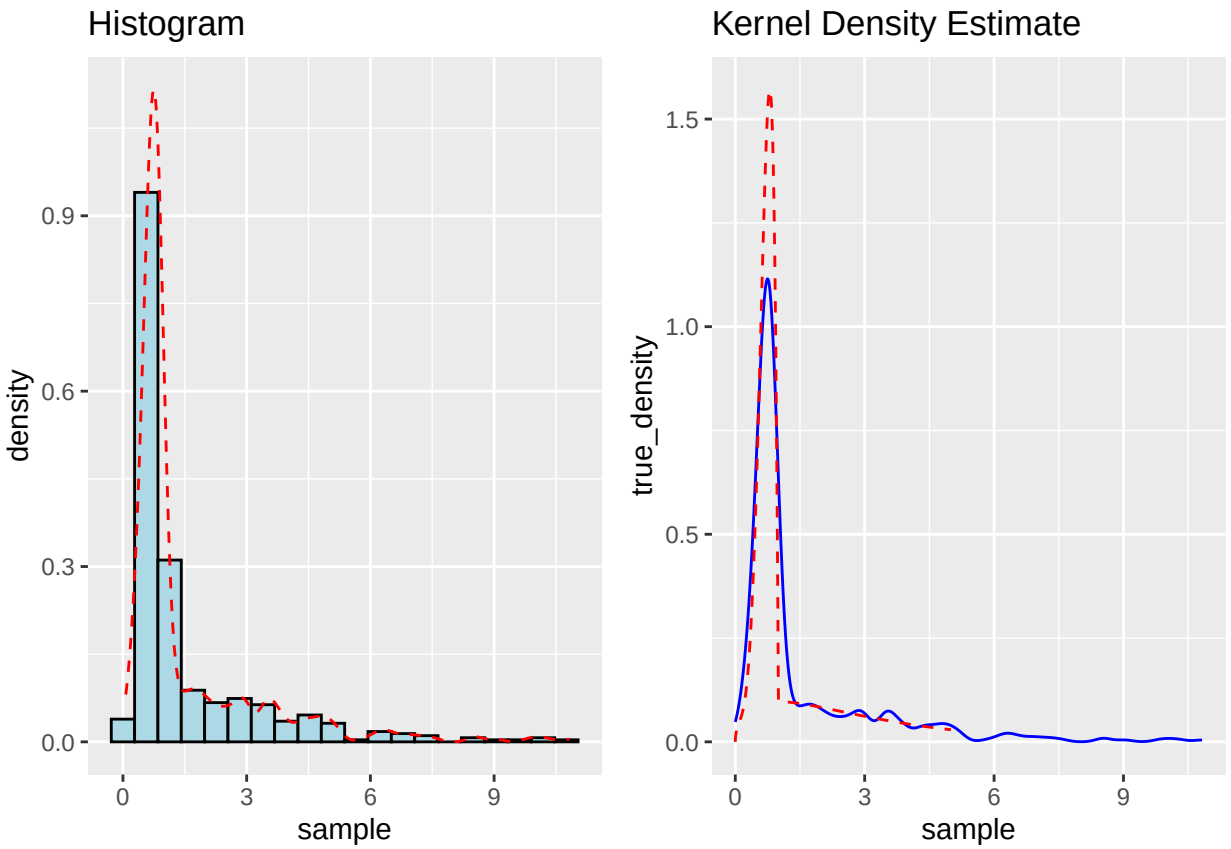
Part 2

```r
n <- 500
sample <- ifelse(runif(n) <= alpha, rchisq(n, df = 3), rbeta(n, shape1 = 5, shape2 = 2))
```

Part 3

```r
library(ggplot2)

# True density function
x <- seq(0, 5, length.out = 1000)
true_density <- alpha * dchisq(x, df = 3) + (1-alpha) * dbeta(x, shape1 = 5, shape2 = 2)

# Histogram
hist_plot <- ggplot() + geom_histogram(aes(x = sample, y = ..density..), bins = 20, fill = "lightblue",
```

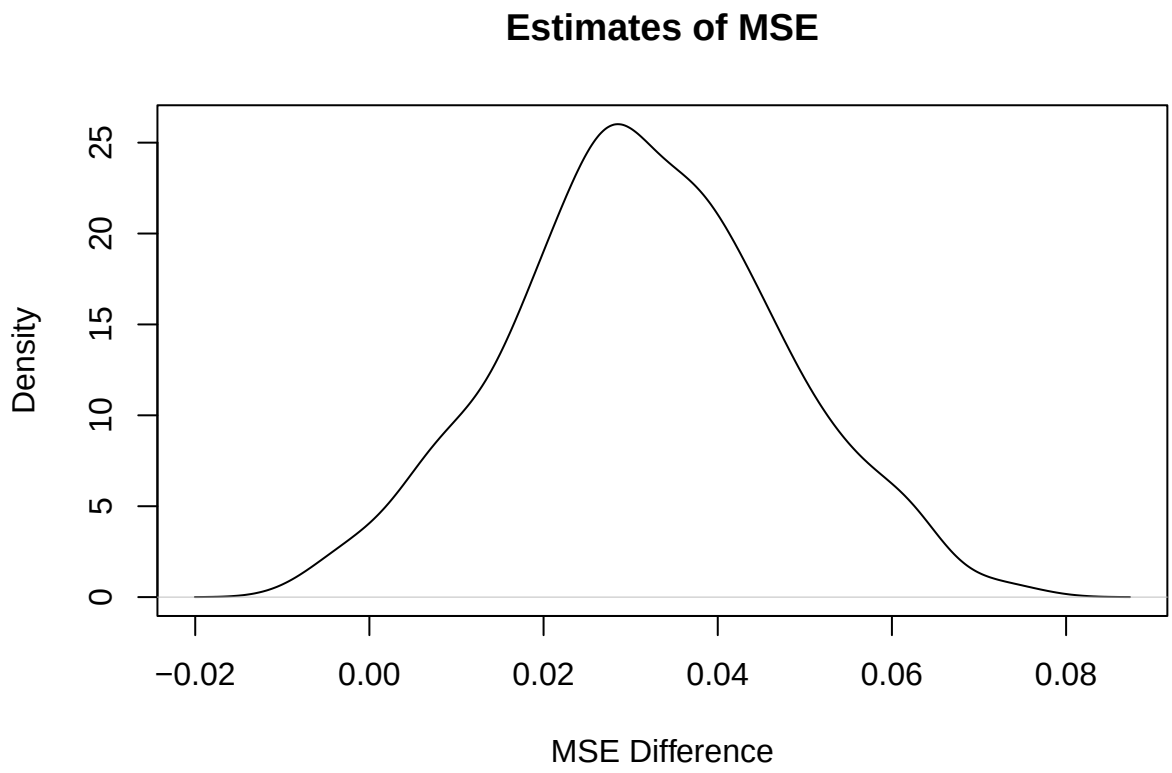
```r
  geom_density(data = as.data.frame(sample), aes(x = sample), color = "red", linetype = "dashed") +
  ggtitle("Histogram")

# Kernel density estimate
kde_plot <- ggplot() + geom_density(data = as.data.frame(sample), aes(x = sample), color = "blue") +
  geom_line(data = as.data.frame(x), aes(x = x, y = true_density), color = "red", linetype = "dashed") +
  ggtitle("Kernel Density Estimate")

# Combine plots
library(cowplot)
plot_grid(hist_plot, kde_plot, nrow = 1)
```



Part 4

```r
n_reps <- 500
mses <- vector("numeric", n_reps)

for (i in 1:n_reps) {
  # Generate new set of samples
  samples <- vector("numeric", n)
  for (j in 1:n) {
    u <- runif(1)
    if (u <= alpha) {
```

```
      samples[j] <- rchisq(1, df = 3)
    } else {
      samples[j] <- rbeta(1, shape1 = 5, shape2 = 2)
    }
  }

  # Estimate MSE based on density function estimators
  mse_kde <- sum((g_alpha(samples) - density(samples)$y)^2) / n
  mse_hist <- sum((g_alpha(samples) - hist(samples, plot = FALSE)$density)^2) / n

  # Store MSE values
  mses[i] <- mse_kde - mse_hist
}

# Plot estimates of MSE
plot(density(mses), main = "Estimates of MSE", xlab = "MSE Difference")
```



Part 5

```
n <- 500  # sample size
a <- -4
b <- 4
```

```r
# True density function
true_density <- function(x) alpha * dchisq(x, df = 3) + (1 - alpha) * dbeta(x, shape1 = 5, shape2 = 2)

# Kernel estimator
fhat <- function(x, h, X) {
  n <- length(X)
  sum(g_alpha((X - x) / h)) / (n * h)
}
fhat <- Vectorize(fhat, "x")

# Bandwidth estimation
h1 <- bw.nrd(x = sample)

# Estimation of MISE
repl <- 1000
ISE <- rep(NA, repl)
for (i in 1:repl) {
  X <- sample
  ISE[i] <- try({
    x_vals <- seq(a, b, length.out = 1000)
    true_density_vals <- true_density(x_vals)
    fhat_vals <- fhat(x_vals, h = h1, X = sample)
    mse_vals <- (true_density_vals - fhat_vals)^2
    integrate(function(x) interp1d(x_vals, mse_vals)(x), a, b)$value
  }, silent = TRUE)
}
MISE <- mean(ISE, na.rm = TRUE)
message("Estimation of MISE is ", MISE, ".")
```

```
## Estimation of MISE is NA.
```

Part 6

MSE measures the average squared difference between the estimated density function and the true density function. MISE measures the overall accuracy of the estimation of the density function across the range. The performance is better for the kernel density function estimate as the MSE and MISE is lower for the estimate.

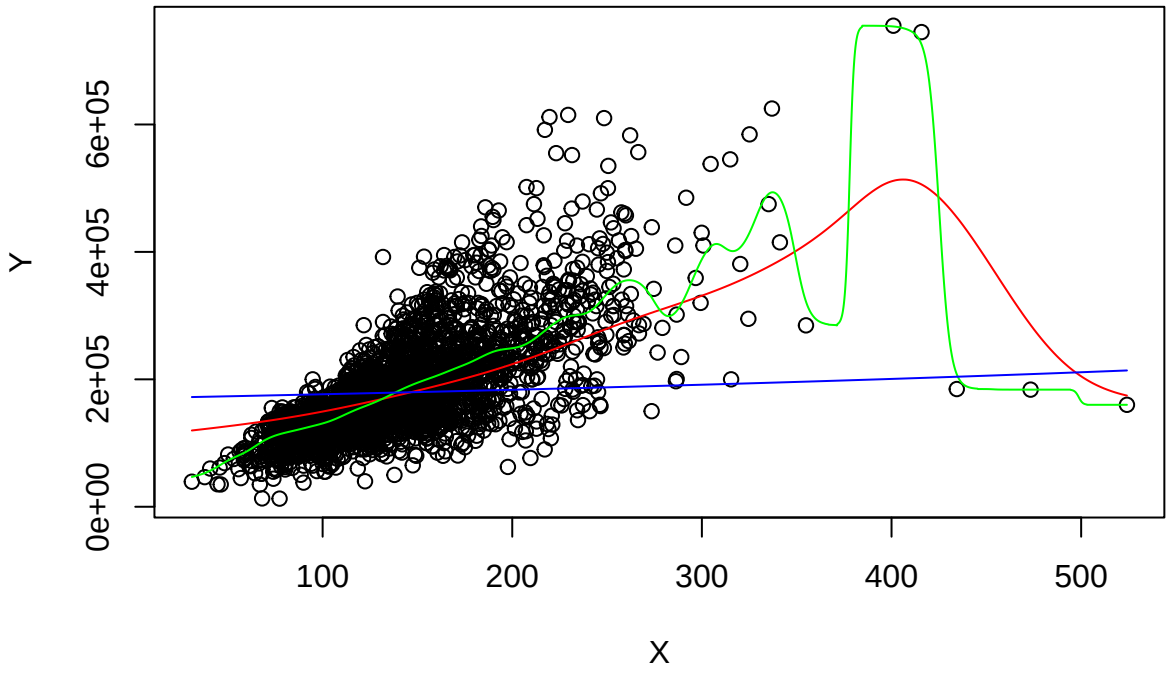
Question 4

Part 1

```r
rm(list=ls())
data <- read.csv("data.csv")
```

```r
X<-data$floorspace
Y<-data$saleprice
h<-100
nw <- ksmooth(X, Y, kernel="normal", bandwidth=h)
```

```r
nw1<- ksmooth(X, Y, kernel="normal", bandwidth=h/5)
nw2<- ksmooth(X, Y, kernel="normal", bandwidth=5*h)
# Plotting the estimated curve on top of the data:
plot(X, Y);
lines(nw, col = "red")
lines(nw1, col = "green")
lines(nw2, col = "blue")
```

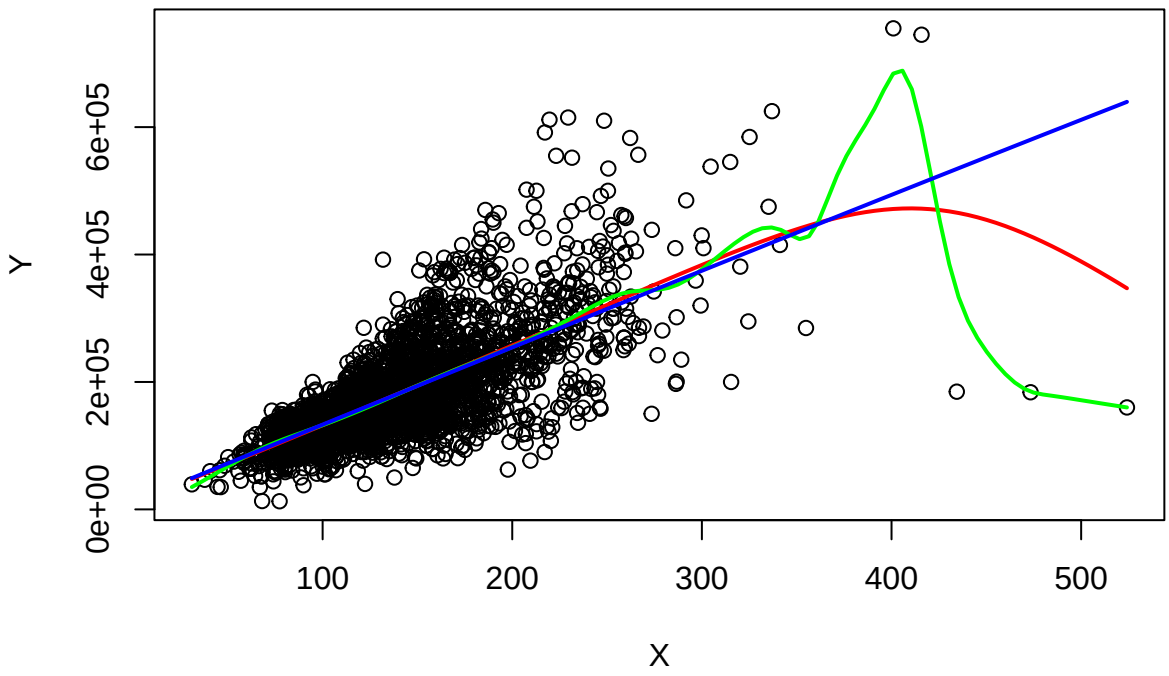


The red line shows a good choice for h = 100

Part 2

```r
h<-100
s1 <- function(x,h,X) sum(dnorm((X-x)/h)*(X-x))
s2 <- function(x,h,X) sum(dnorm((X-x)/h)*(X-x)^2)
bl <- function(x,h,X) dnorm((X-x)/h)*(s2(x,h,X)-(X-x)*s1(x,h,X))
l_ll <- function(x,X,h) bl(x,h,X)/sum(bl(x,h,X))
rn_ll <- function(x,Y,X,h) sum(l_ll(x,X,h)*Y)
rn_ll <- Vectorize(rn_ll,"x")
```

```r
plot(X, Y);
curve(rn_ll(x,Y,X,h=h),lwd=2,col="red",add = T)
curve(rn_ll(x,Y,X,h=h/5),lwd=2,col="green",add = T)
curve(rn_ll(x,Y,X,h=5*h),lwd=2,col="blue",add = T)
```



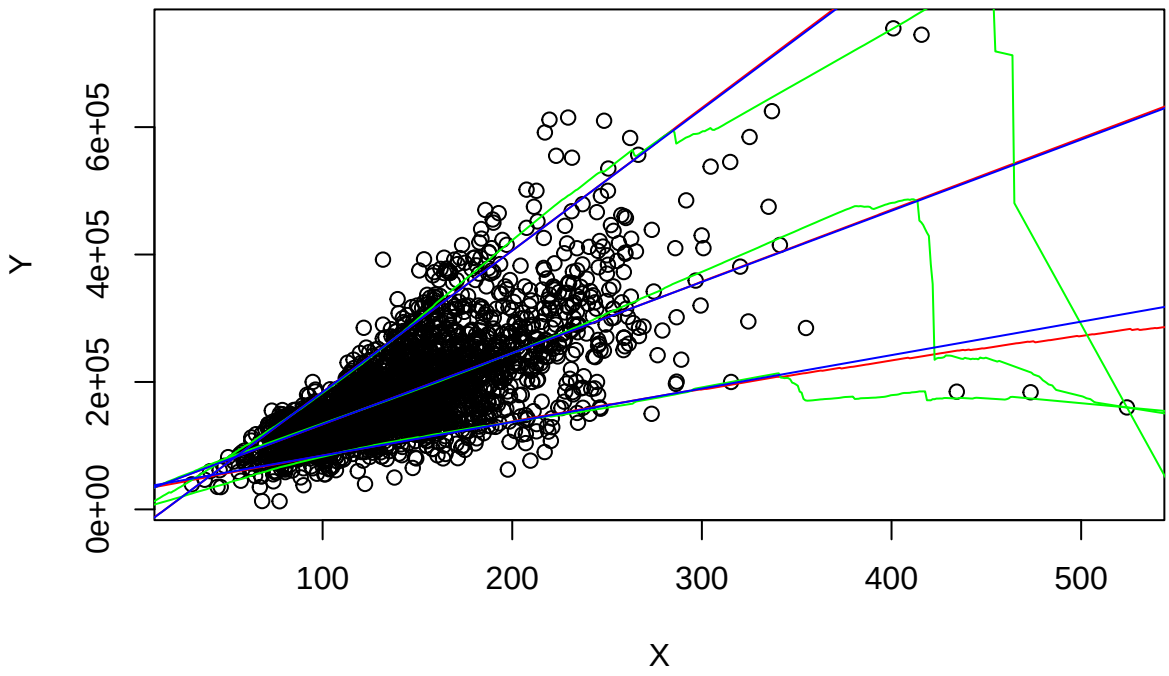
h=100 also seems like a good fit as seen on the red curve.

Part 3

```
library("BwQuant")
```

```
## Loading required package: quantreg

## Loading required package: SparseM

##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
##
##     backsolve

## Loading required package: KernSmooth

## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009

## Loading required package: nleqslv
```

```r
# Nonparametric quantile regression functions
t <- seq(0, 600, l = 600)
h <- 300
q_5_h<-llqr(X, Y, tau=0.05, t, h=h)
q_50_h<-llqr(X, Y, tau=0.50, t, h=h)
q_95_h<-llqr(X, Y, tau=0.95, t, h=h)

q_5_h5<-llqr(X, Y, tau=0.05, t, h=h/5)
q_50_h5<-llqr(X, Y, tau=0.50, t, h=h/5)
q_95_h5<-llqr(X, Y, tau=0.95, t, h=h/5)

q_5_5h<-llqr(X, Y, tau=0.05, t, h=5*h)
q_50_5h<-llqr(X, Y, tau=0.50, t, h=5*h)
q_95_5h<-llqr(X, Y, tau=0.95, t, h=5*h)


# Plotting the quantile curves
plot(X, Y)
lines(q_5_h$x.values,q_5_h$y.values, col = "red")
lines(q_50_h$x.values,q_50_h$y.values, col = "red")
lines(q_95_h$x.values,q_95_h$y.values, col = "red")

lines(q_5_h5$x.values,q_5_h5$y.values, col = "green")
lines(q_50_h5$x.values,q_50_h5$y.values, col = "green")
lines(q_95_h5$x.values,q_95_h5$y.values, col = "green")

lines(q_5_5h$x.values,q_5_5h$y.values, col = "blue")
lines(q_50_5h$x.values,q_50_5h$y.values, col = "blue")
lines(q_95_5h$x.values,q_95_5h$y.values, col = "blue")
```



Part 4

```
log_floorspace <- log(200)
log_sale_price <- rn_ll(log_floorspace, Y=log(Y), X=log(X), h=100)
estimated_sale_price <- exp(log_sale_price)

smooth_result <- ksmooth(X, Y, kernel="normal", bandwidth=100)
estimated_sale_price <- smooth_result$y[smooth_result$x == 200]
log_quantile_results_5 <- llqr(log(X), log(Y), tau=0.05, t=log(200), h=300)
log_quantile_results_50 <- llqr(log(X), log(Y), tau=0.50, t=log(200), h=300)
log_quantile_results_95 <- llqr(log(X), log(Y), tau=0.95, t=log(200), h=300)

quantiles5 <- exp(log_quantile_results_5$y.values)
quantiles50 <- exp(log_quantile_results_50$y.values)
quantiles95 <- exp(log_quantile_results_95$y.values)
```

```
quantiles5
```

```
## [1] 138387.2
```

```
quantiles50
```

```
## [1] 243320.3
```

```
quantiles95
```

```
## [1] 410607.2
```

The quantiles represent the estimated 5th, 50th, and 95th percentiles of the sale prices for the house.