---
title: "36-402 Homework 9"
author: "Jacky Liu"
date: "2/12/21"
output: pdf_document

Problem 1

Models:

```
abalone = read.csv("abalonemt.csv")
library(np)
```

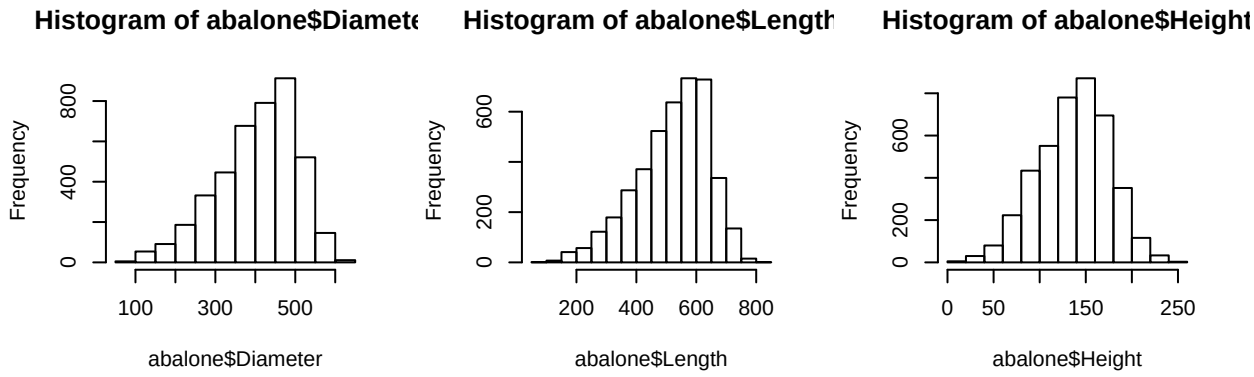
```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-10)
## [vignette("np_faq",package="np") provides answers to frequently asked questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

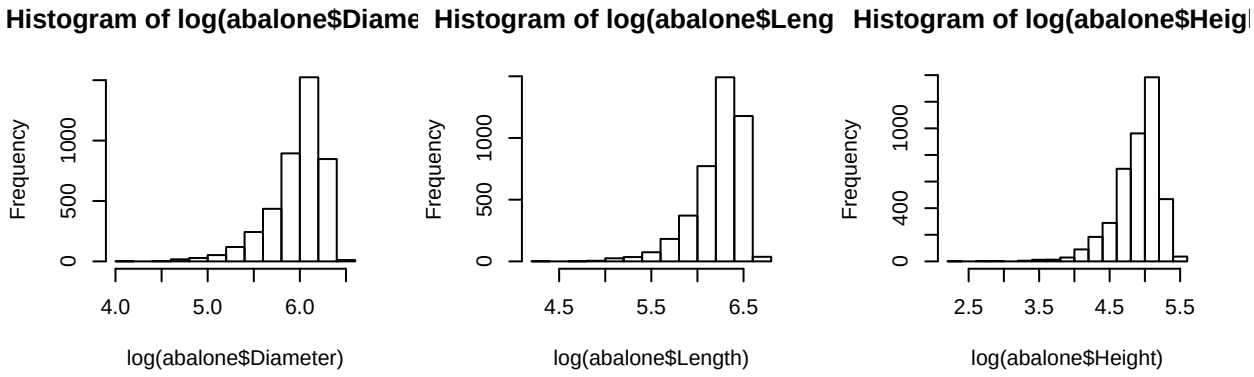
```
model1 = lm(log(Shucked.weight) ~ log(Diameter) + log(Length) + log(Height), data=abalone)
bw <- apply(abalone[,c(3,2,4)], 2, sd) / nrow(abalone)^(0.2)
model2 = npreg(Shucked.weight ~ Diameter + Length + Height, bws = bw, data = abalone, residuals = TRUE)
model3 = smooth.spline(abalone$Shucked.weight ~ abalone$Diameter * abalone$Length * abalone$Height)
```

Problem 1 (a)

1. Univariate Plots:

```
par(mfrow = c(2, 3))
hist(abalone$Diameter)
hist(abalone$Length)
hist(abalone$Height)
hist(log(abalone$Diameter))
hist(log(abalone$Length))
hist(log(abalone$Height))
```

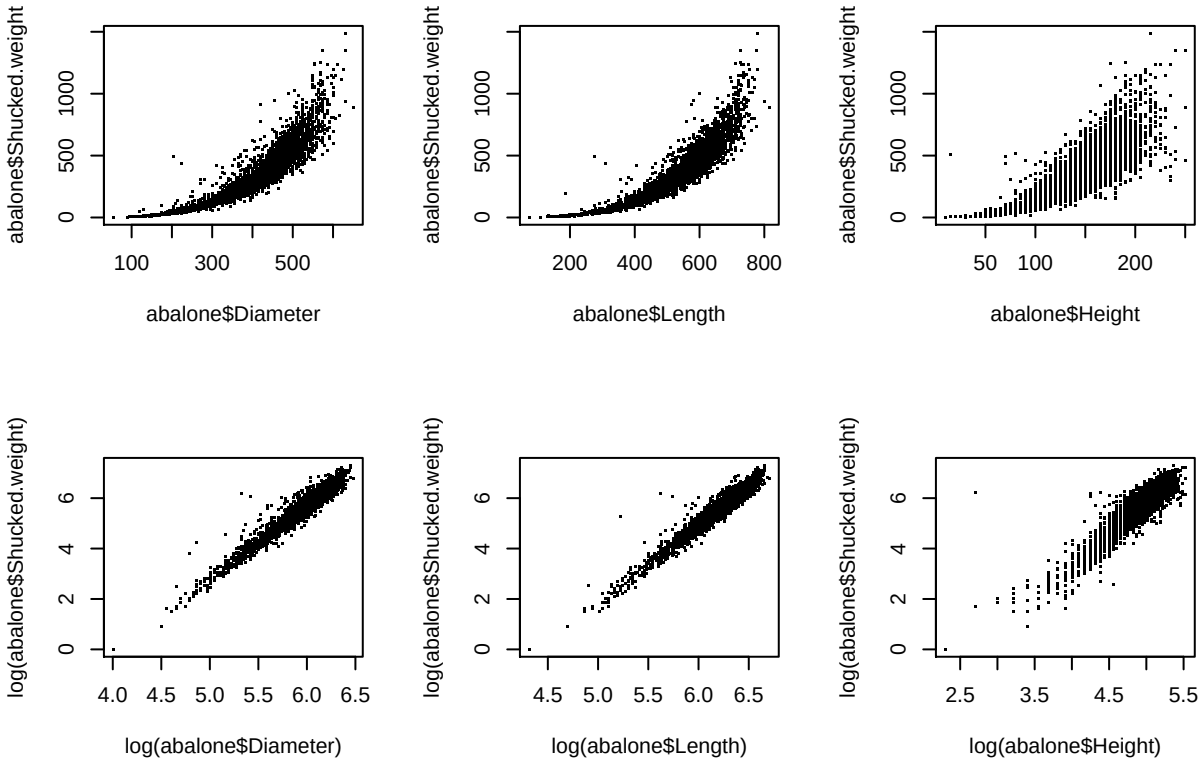




All three of the logged marginal distributions of the predictor variables seem more left-skew than the original. Therefore, the log transform seems unnecessary.

2: Bivariate plots

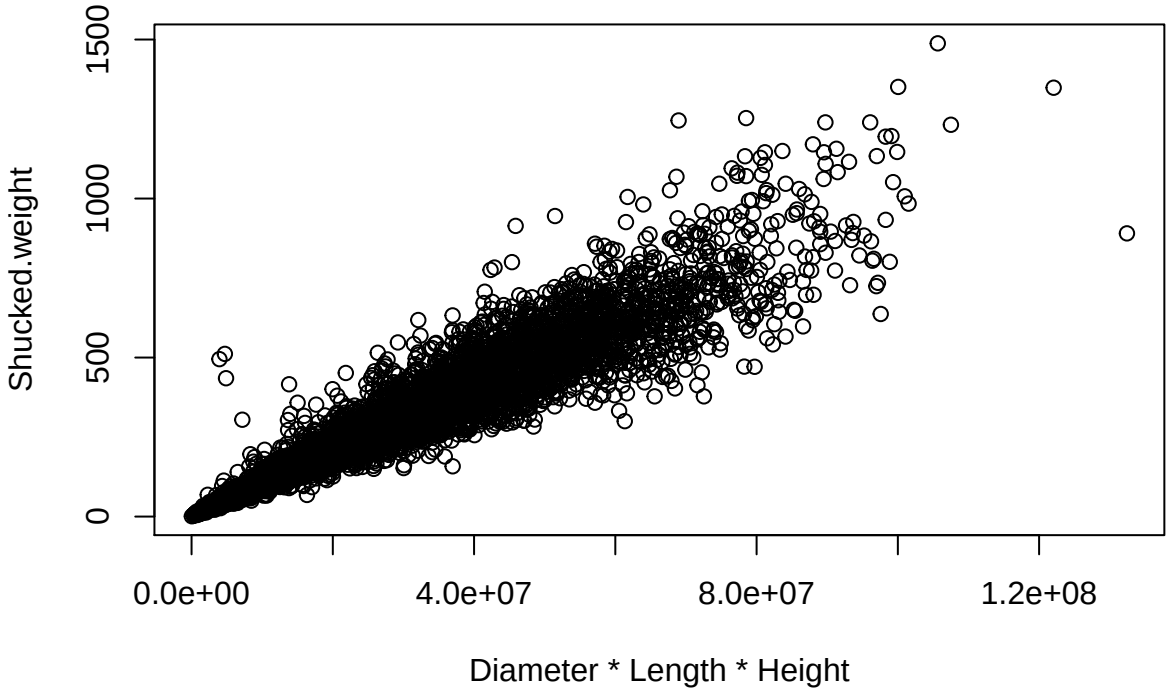
```
par(mfrow = c(2, 3))
plot(abalone$Diameter, abalone$Shucked.weight, pch=".")
plot(abalone$Length, abalone$Shucked.weight, pch=".")
plot(abalone$Height, abalone$Shucked.weight, pch=".")
plot(log(abalone$Diameter), log(abalone$Shucked.weight), pch=".")
plot(log(abalone$Length), log(abalone$Shucked.weight), pch=".")
plot(log(abalone$Height), log(abalone$Shucked.weight), pch=".")
```



The linearity assumption for model 1 seems to be violated. The log transformation improves the relationship and seems much more linear.

3: Plot of Shucked.weight against the product of three predictors:

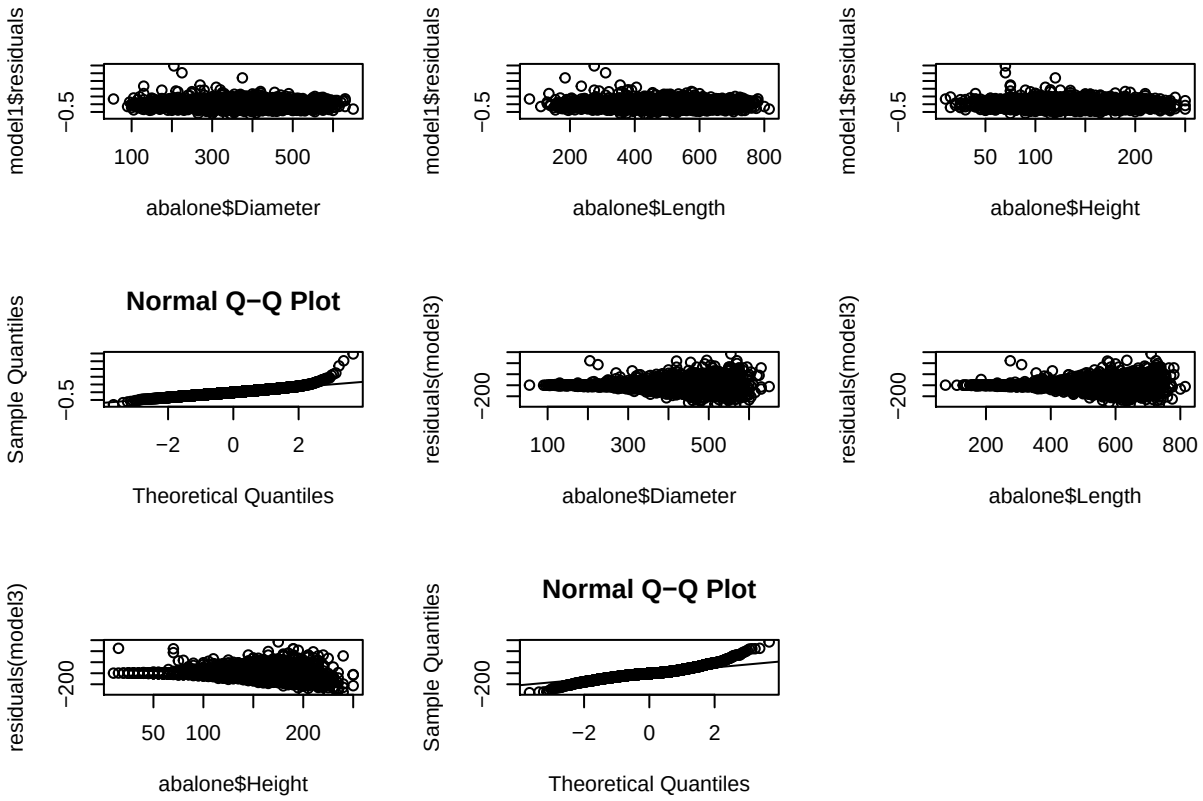
```
with(abalone, plot(Diameter*Length*Height, Shucked.weight))
```



The relationship between Shucked.weight and the product of the three predictors seem fairly linear. We would expect Model 3 to fit the data well, but not necessarily as well as a linear model.

Problem 1 (b)

```
par(mfrow = c(3, 3))
plot(abalone$Diameter, model1$residuals)
plot(abalone$Length, model1$residuals)
plot(abalone$Height, model1$residuals)
qqnorm(model1$residuals); qqline(model1$residuals)
plot(abalone$Diameter, residuals(model3))
plot(abalone$Length, residuals(model3))
plot(abalone$Height, residuals(model3))
qqnorm(residuals(model3)); qqline(residuals(model3))
```



The residuals for model 1 indicate that the assumptions of the linear model (iid, linearity, constant variance, Gaussian noise) seem relatively but not perfectly plausible. Although the points seem to be evenly distributed with constant variance, the tails seem to drift off at the ends in the Q-Q plot. Overall the data show some modest signs of betraying the linear model assumptions, but not too extreme.

The residual plots for model 3 show both change in slope and change in variance. This suggests that the relationship might now be linear. For this reason, it is better to resample (X,Y) pairs bootstrap instead of drawing bootstrap samples from a distribution with Y=r(x) plus noise.

Problem 1 (c)

```
options(scipen=999)
set.seed(69)

B = 1000
n = nrow(abalone)

newdata <- c(50, 340, 465, 490, 750) * c(80, 475, 525, 655, 875) * c(12, 110, 142, 156, 450)
#predict(model3, newdata)

predictions = list()

for (j in 1:B) {
  therows <- sample(n, n, replace=T)
  datab <- data.frame(Diameter = abalone$Diameter[therows],
```

```
                      Length = abalone$Length[therows],
                      Height = abalone$Height[therows],
                      Shucked.weight = abalone$Shucked.weight[therows])
  modelb = smooth.spline(datab$Shucked.weight ~ datab$Diameter * datab$Length * datab$Height)
  #print(predict(modelb, newdata)$y)
  predictions[[j]] = predict(modelb, newdata)$y
}
preds = matrix(unlist(predictions), nrow=5, ncol=B)
rx = apply(preds, 1, mean)
preds.sorted = t(apply(preds, 1, sort))
confint.lower = apply(preds.sorted, 1, function(x) x[25])
confint.upper = apply(preds.sorted, 1, function(x) x[975])
cat("Means:", rx, "\n")
```

```
## Means: 1.960594 195.6584 364.5167 511.3318 -16016.98
```

```
cat("Lower Bounds:", confint.lower, "\n")
```

```
## Lower Bounds: -0.1712706 186.7074 347.9788 492.7372 -46345.21
```

```
cat("Upper Bounds:", confint.upper, "\n")
```

```
## Upper Bounds: 3.577799 206.4347 384.35 529.4438 20075.61
```

Length	Diameter	Height	r^(x)	95% CI of r(x)
80	50	12	1.960594	(-0.1712706, 3.577799)
475	340	110	195.6584	(186.7074, 206.4347)
525	465	142	364.5167	(347.9788, 347.9788)
655	490	156	511.3318	(492.7372, 529.4438)
875	750	450	-16016.98	(-46345.21, 20075.61)