Homework 6

Advanced Methods for Data Analysis (36-402)

Due **Thursday March 3**, 2022 at **8:00 pm ET**

Solutions – not to be posted online or shared, even after the end of the semester.

You should **always show all your work** and submit both a writeup and *R* code.

- Assignments must be submitted through Gradescope as a PDF. Follow the instructions here: `https://www.cmu.edu/teaching/gradescope/`
- Gradescope will ask you to mark which parts of your submission correspond to each homework problem. This is mandatory; if you do not, grading will be slowed down, and your assignment will be penalized.
- Make sure your work is legible in Gradescope. You may not receive credit for work the TAs cannot read. **Note:** If you submit a PDF with pages much larger than 8.5×11", they will be blurry and unreadable in Gradescope.
- For questions involving R code, we strongly recommend using R Markdown. The relevant code should be included with each question, rather than in an appendix. A template Rmd file is provided on Canvas.

1. **The Revenge of the Housing Data.** In Homework 5, we compared four models for predicting median house value from various predictors. Two of the models had very similar prediction errors, and it wasn't obvious that one model was better than the other. The bootstrap allows us to measure the uncertainty in the prediction (cross-validation) mean-squared-error calculations to help decide whether one of the models is actually better than the other.

   (a) Read the two data files `housetrain.csv` and `housetest.csv` back into R and combine them into a single object, as you did in Homework 5. This time, we are going to bootstrap the 10-fold cross-validation analysis for the models that were called Model 3 and Model 5 in Homework 5:

   Model 3: A multiple regression of `Median_house_value` on `Median_household_income` and `Mean_household_inco`

   Model 5: A kernel regression of `Median_house_value` on `Median_household_income` and `Mean_household_income`

   Some of the code you used for cross-validation in previous assignments, with modifications, can be useful for this problem.

   Create $B = 200$ bootstrap samples, each consisting of $n = 10605$ rows from the combined data set selected at random with replacement. For each bootstrap sample b, randomly divide the n observations in to 10 disjoint sets. Treating one fold as test data and the other 9 as training data, calculate the prediction error for each model, and repeat this using each fold in turn as the training data. Call the average of the 10 prediction errors for Model 3 $\widehat{\text{MSE}}^*_{3b}$. Call the average for Model 5 $\widehat{\text{MSE}}^*_{5b}$. You should have 200 of each of these.

   Draw a histogram of the $\widehat{\text{MSE}}^*_{3b} - \widehat{\text{MSE}}^*_{5b}$ values. What **visual** evidence is there about whether one model is better?

Solution: First, we read and merge the data:

```
suppressMessages(library(np))

housetrain <- read.csv("housetrain.csv", header=TRUE)
housetest <- read.csv("housetest.csv", header = TRUE)

housedata <- rbind(housetrain, housetest)
```

Next, we set up the problem:

```
B <- 200 # number of bootstraps
K <- 10 # number of CV folds
N <- nrow(housedata)

# Divide the 10605 indices into K equal-sized sets
orig <- rep(1:K, length.out = N)

# Make a place to store the differences
pred <- numeric(B)
```

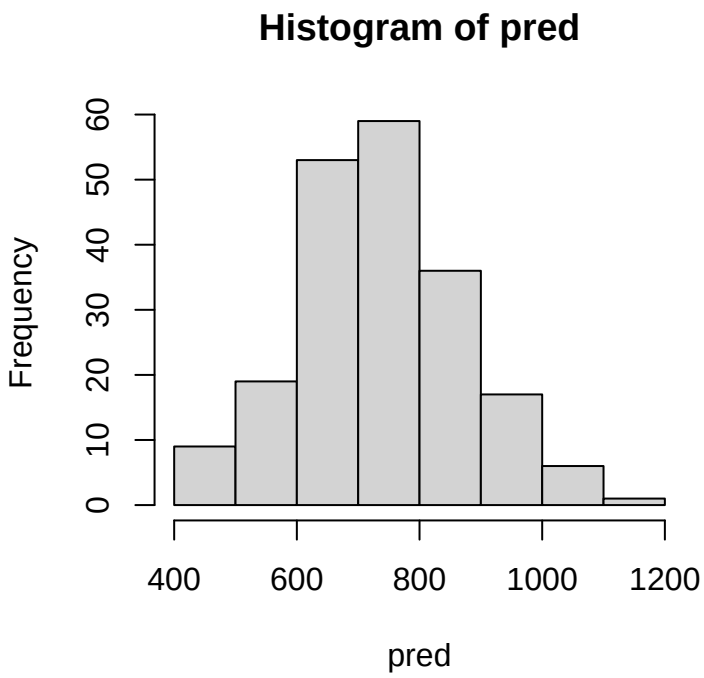
Finally, we conduct the bootstrap and cross-validate inside the bootstrap loop:

```
# Loop through the bootstrap samples
for (b in 1:B) {
  # Choose a new set of 10 folds for each bootstrap sample
  samp <- sample(orig)

  # Choose the bootstrap sample
  boots <- sample(N, N, replace = TRUE)
  tempdata <- housedata[boots, ]

  # Make a place to save the cross-validation errors
  prederr3 <- numeric(K)
  prederr5 <- numeric(K)

  # Loop through the K folds
  for (j in 1:K) {
    testd <- tempdata[samp == j, ]
    traind <- tempdata[samp != j, ]

    model3 <- lm(Median_house_value ~ Mean_household_income +
                   Median_household_income, data = traind)
    prederr3[j] <- mean((predict(model3, newdata = testd)
      - testd$Median_house_value)^2)

    bw <- apply(traind[, 5:6], 2, sd) / nrow(traind)^(0.2)

    model5 <- npreg(Median_house_value ~ Median_household_income +
                      Mean_household_income, data = traind,
                    newdata = testd, bws = bw)
    prederr5[j] <- mean((model5$mean - testd$Median_house_value)^2)
  }

  # Take the difference between the two average test errors
  pred[b] <- mean(prederr3 - prederr5)
}
```

Supplying the `bws` argument to `npreg` is crucial here: without it, `npreg` will try to determine the best bandwidth itself, by doing its own cross-validation. And doing cross-validation inside cross-validation can take a long time.

The histogram:

```
hist(pred)
```



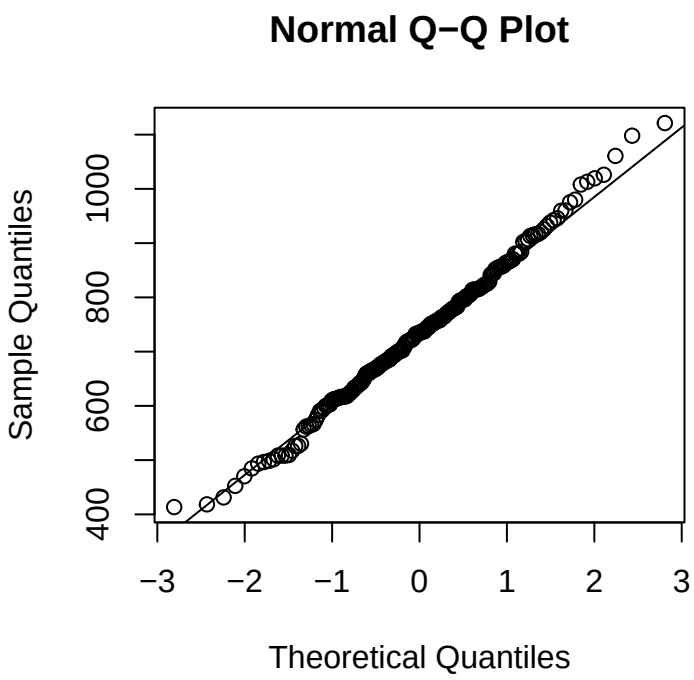
Typically, model 3's error is about 700–800 units higher than model 5's error, implying that model 5 is better. Notice that on none of the 200 iterations did model 3 perform better (lower error).

(b) Let $T_b^* = \widehat{\text{MSE}}_{3b}^* - \widehat{\text{MSE}}_{5b}^*$ for $b = 1, \ldots, 200$. Draw a normal Q-Q plot of the T_b^* values and add the `qqline`. Do they look like a sample of normal random variables?

Regardless of your result, treat $T_1^*, \ldots, T_{200}^*$ as a random sample of normal random variables and test the null hypothesis that $\mathbb{E}[T_j^*] = 0$. Does one of the models look better than the other now?

Solution: Here's the Q-Q plot:

```
qqnorm(pred)
qqline(pred)
```



This looks pretty close to normally distributed.

We conduct a t-test to test the null hypothesis.

```
t.test(pred)
##
##  One Sample t-test
##
## data:  pred
## t = 76.786, df = 199, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  714.7151 752.3921
## sample estimates:
## mean of x
##  733.5536
```

The t test rejects the null hypothesis that the two models have the same error, so it appears Model 5 really is better than Model 3.

2. **Abalone sizes.** Abalone are a type of sea snail often harvested and sold for food. The data file `abalonemt.csv` contains nine variables measured on 4173 abalones. Here is a description of the nine variables:

Name	Data Type	Units	Description
Sex	nominal	M, F, I (infant)	
Length	continuous	mm	
Diameter	continuous	mm	
Height	continuous	mm	
Whole.weight	continuous	grams	Entire abalone
Shucked.weight	continuous	grams	Weight of meat
Viscera.weight	continuous	grams	gut weight (after bleeding)
Shell.weight	continuous	grams	after being dried
Rings	integer		+1.5 gives the age in years

A fisherperson who wishes to sell the abalone for food is interested in the edible part. Prior to cutting one open for cooking, one can predict the weight of the edible part (`Shucked.weight`) through a regression model. A natural predictor that would be available, if a scale were available, is `Whole.weight`.

Assume that the only predictors available, when the prediction of `Shucked.weight` is needed, are `Diameter`, `Length`, and `Height`.

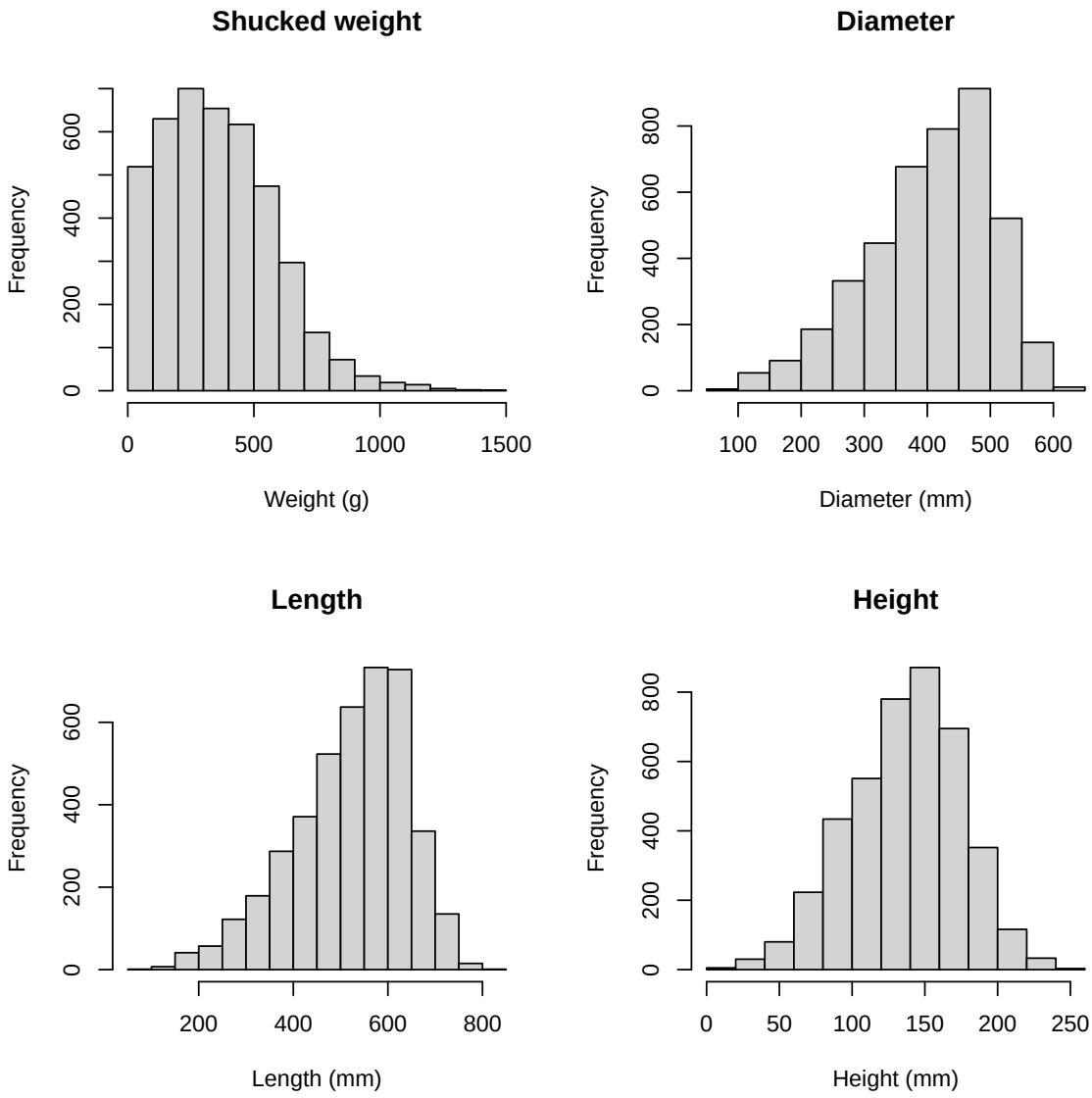
Here is one hint: Objects of uniform density have weights proportional to their volume. What functions of the predictors might be good predictors of the volume of an abalone?

(a) As usual, start with an exploratory data analysis (EDA) of the response, `Shucked.weight`, and the three predictors: `Diameter`, `Length`, and `Height`. Think of this as practice for the Data Exam: Conduct EDA as though you are preparing to build models of these variables and want to determine how you should include them in the model and what relationships to expect.

Solution: A good EDA should help us understand the data. What do the variables look like, and what relationships are visible between the predictors and response?

Examining the variables individually, we see the following distributions:

```
abalone <- read.csv("abalonemt.csv", header = TRUE)

par(mfrow = c(2, 2))

hist(abalone$Shucked.weight, xlab = "Weight (g)",
     main = "Shucked weight")

hist(abalone$Diameter, xlab = "Diameter (mm)",
     main = "Diameter")
hist(abalone$Length, xlab = "Length (mm)",
     main = "Length")
hist(abalone$Height, xlab = "Height (mm)",
     main = "Height")
```

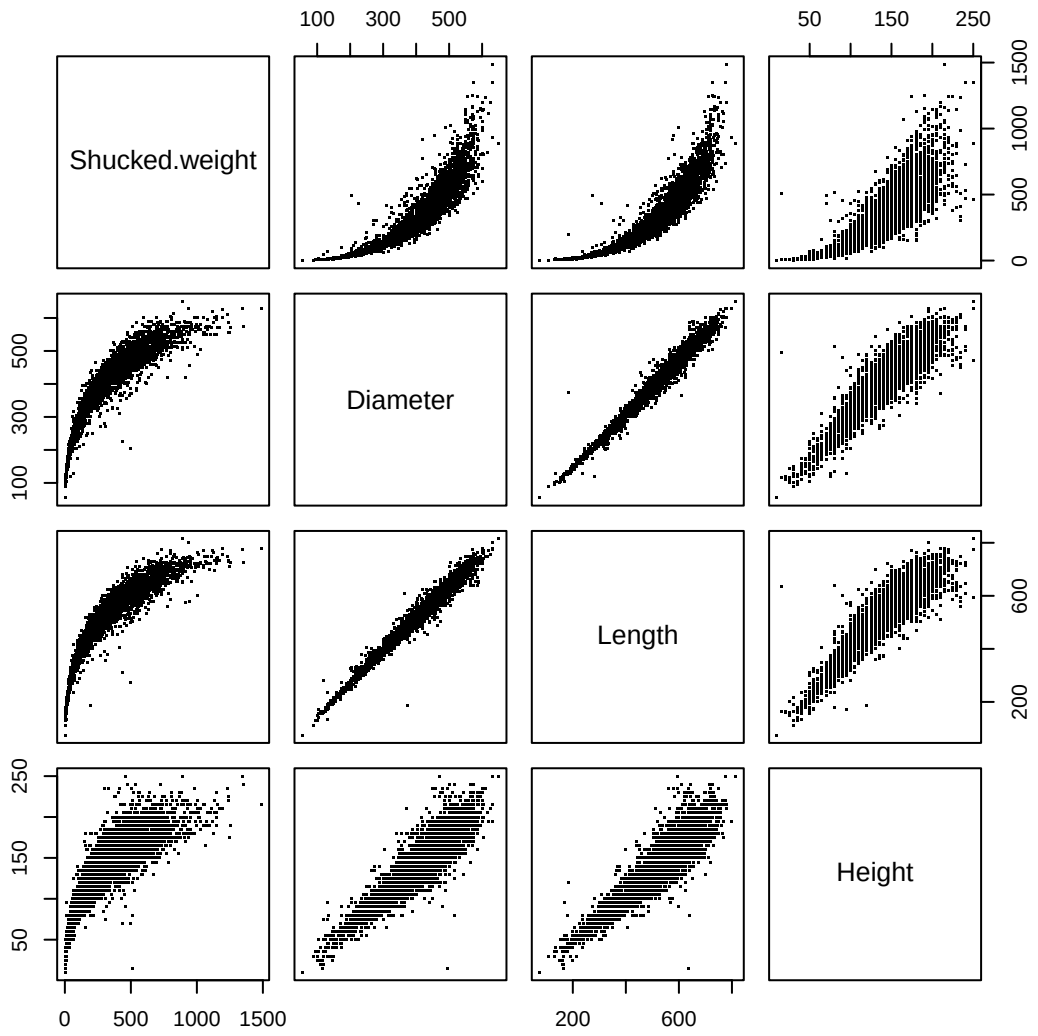


We see that shucked weight has a skewed distribution, as one might expect: there are no weights below 0 grams, the most common weights are below 500 grams, and occasionally there are huge abalone with weights near 1500 grams.

The predictor distributions are all similar to each other. We see nothing abnormal, like lengths smaller than 0 mm. The predictor distributions are skewed, but remember that linear regression does **not** assume that predictors are normally distributed; the only normality assumption is for the residuals ϵ.

We can plot these against each other to see the relationships:

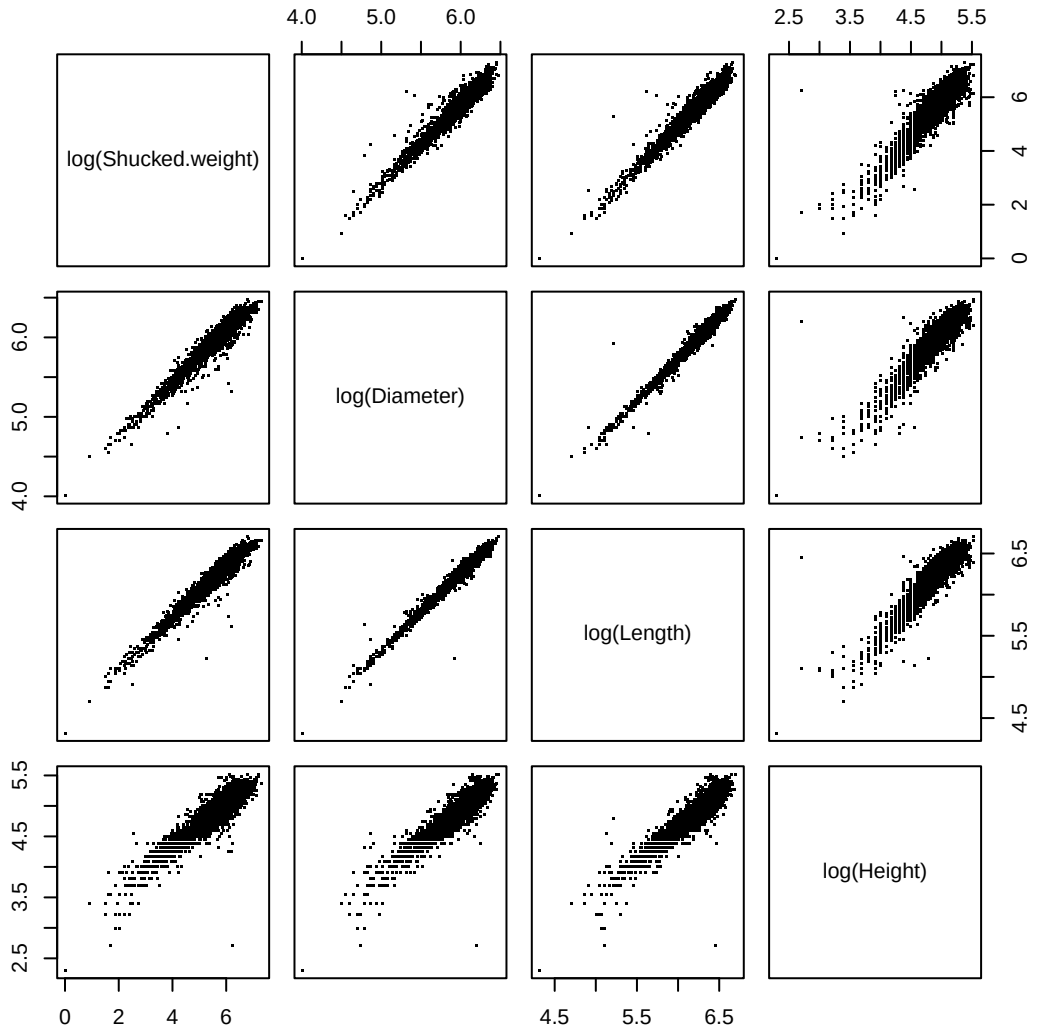
```
pairs(~ Shucked.weight + Diameter + Length + Height,
      data = abalone, pch = ".")
```



Shucked weight has a strong curved relationship with the three dimension variables, which themselves are rather linearly related. Note also how the variance increases as the predictors increase. Using a linear model to predict shucked weight from these predictors would not work well.

We may expect that weight is proportional to length × diameter × height, provided abalone have similar density. Because the log of a product is the sum of the logs, we might expect log of weight to be more linearly related to the logs of the variables:

```
pairs(~ log(Shucked.weight) + log(Diameter) + log(Length) + log(Height),
      data = abalone, pch = ".")
```



The relationships indeed look linear now, though perhaps with increasing variance for larger values of the predictors. Height seems to be less highly correlated with the other three variables than each of the other three are with each other.

This suggests that if we use a linear model, it should be a linear relationship between the logs of all the variables; without the logs, we'd need a nonlinear model.

A few notes about good EDA:

- The plots shown above focus on the variables we are using for this problem, not any other variables in the data. The text focuses on aspects of the data that are important to our models; for instance, we discuss the shape of relationships with shucked weight, but don't emphasize the skew in the distributions of the individual variables, because that skew is not important for our models.
- The plots help us understand the variables and detect problems; the histograms would have revealed any problems like unexpected data values, and the scatterplots show what models may be appropriate. A good EDA will help inform your analysis. We didn't include numerous other plots that show nothing interesting.

- We did not report lots of other numbers that are not relevant. For instance, reporting the mean, standard deviation, median, and quantiles of every variable is not very helpful when we already have histograms, unless those summary statistics reveal something interesting that will change how we do the analysis.
- The histograms all have axes labels and reasonable titles that describe the variables, rather than using the raw variable names in R. (This is more important when you have datasets with column names like "PCTCHG" or "SHCKWGHT"; a reader should be able to understand the plots without needing to understand weird variable names.)

(b) We will explore linear and smoothing spline models for predicting the response from some or all of the predictors. Smoothing spline models are an extension of the spline models we will discuss in class on Tuesday; they can be fit using the `smooth.spline` function, which takes separate `x` and `y` arguments. For example,

```
spline_fit <- smooth.spline(x = abalone$Diameter,
                            y = abalone$Shucked.weight)
```

R will automatically used generalized cross-validation to determine the optimal penalty parameter, which we will discuss in class; for this homework, simply allow R to choose the best parameters. You can make predictions using `predict`, but with slightly different syntax:

```
preds <- predict(spline_fit, x = c(10, 20))
preds$x # will be vector of 10, 20
preds$y # will be vector of predictions for those diameters
```

See the R help for `predict.smooth.spline` for details.

You should fit the following models:

Model 1: A linear regression of log(`Shucked.weight`), on the logarithms of all three predictors.

Model 2: A smoothing spline for predicting `Shucked.weight` from `Diameter` times `Length` times `Height`.

Note: When making predictions with Model 1, the `predict` function will use the predictors correctly, but it will predict the logarithm of the response. You will need to take `exp` of the predictions before comparing the predictions to `Shucked.weight` in the calculation of cross-validation error.

Draw a scatter plot of the fitted values for both models against each other, and comment on how similar or different the predictions seem to be.

Examine residuals from each model and say what you learn about the possible distributions of the noise terms, and comment on how well the usual regression assumptions seem to hold.

Solution: Model 1 looks like this:

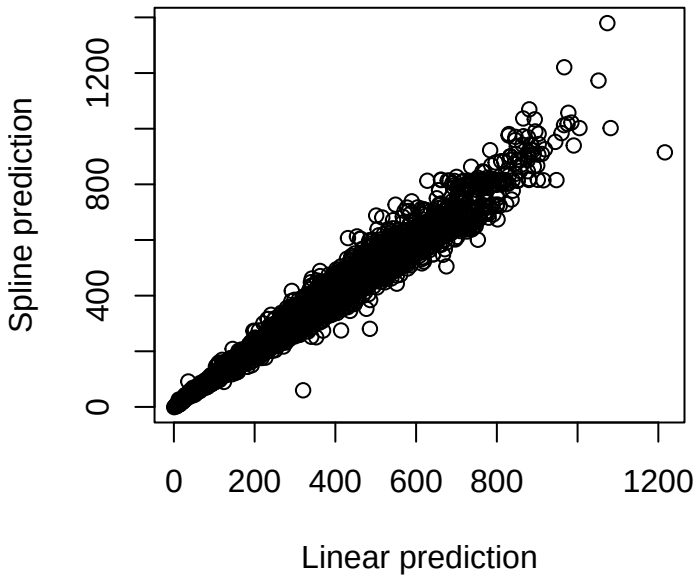
```
model1fit <- lm(log(Shucked.weight) ~ log(Length) + log(Diameter) +
                  log(Height),
                data = abalone)
```

Model 2 uses a smoothing spline:

```
model2fit <- smooth.spline(y = abalone$Shucked.weight,
                           x = abalone$Length * abalone$Diameter *
                             abalone$Height)
```

We plot the predictions against each other:

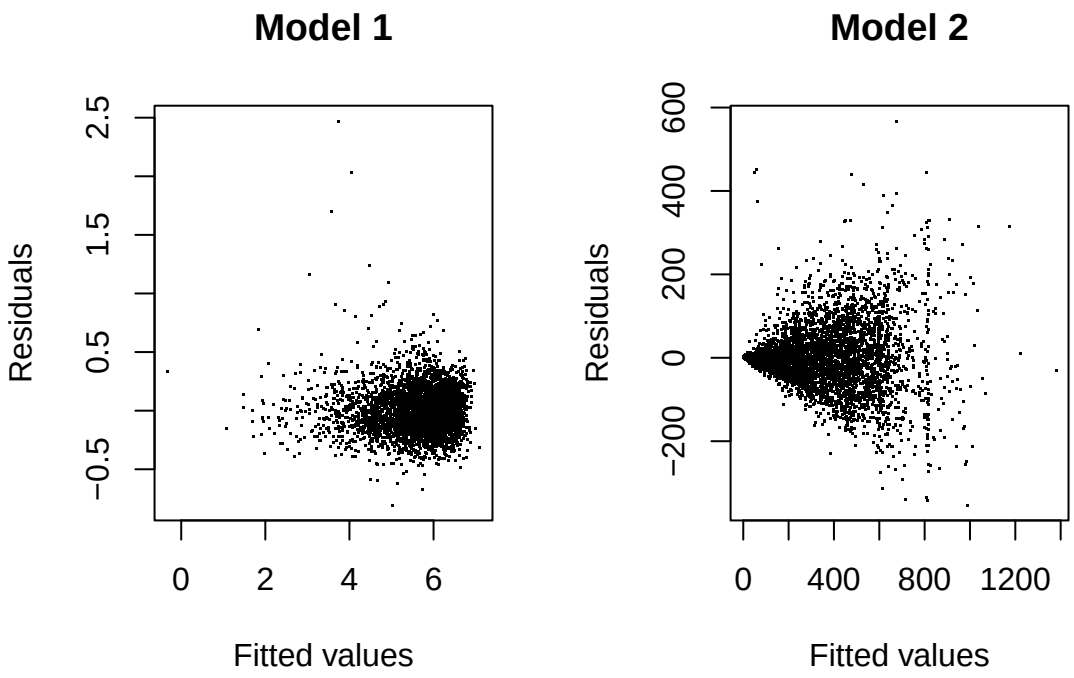
```
thefits <- data.frame(LinearLog = exp(fitted(model1fit)),
                      Spline = fitted(model2fit))
plot(Spline ~ LinearLog, data = thefits,
     xlab = "Linear prediction", ylab = "Spline prediction")
```



The predictions seem to be similar; they're most similar for abalone predicted to have small weights, and more different for abalone predicted to have large weights.

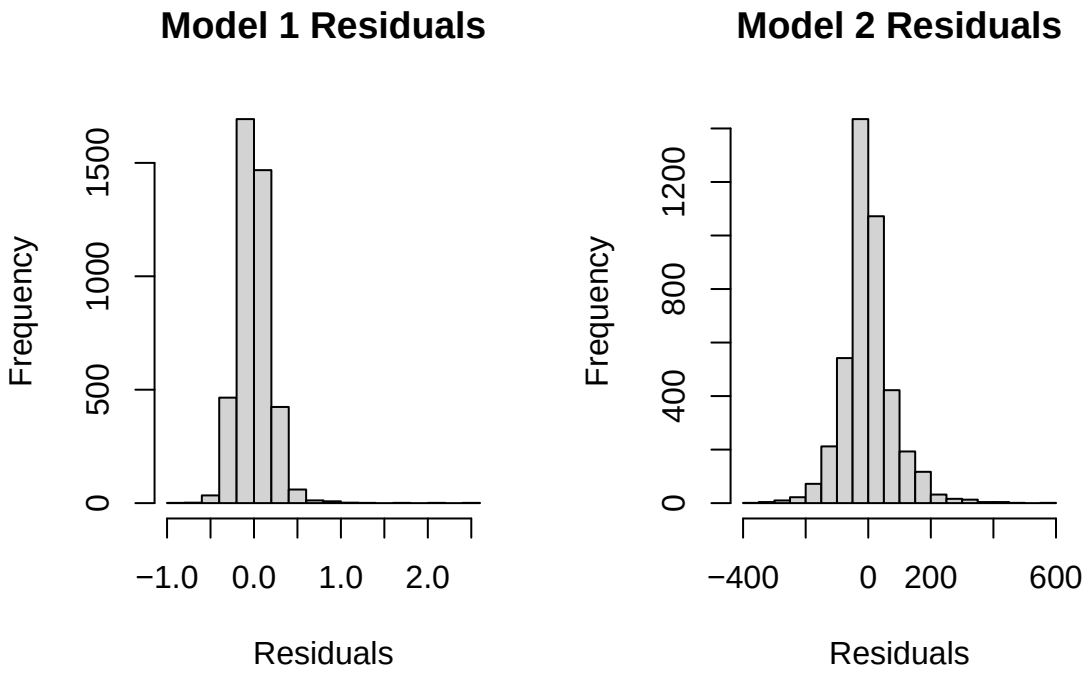
Let's examine the residuals next:

```
par(mfrow = c(1, 2))
plot(fitted(model1fit), residuals(model1fit), pch=".", xlab="Fitted values",
     ylab="Residuals", main="Model 1")

plot(fitted(model2fit), residuals(model2fit), pch=".", xlab="Fitted values",
     ylab="Residuals", main="Model 2")
```



And their distributions:

```
par(mfrow=c(1,2))
hist(residuals(model1fit), main="Model 1 Residuals", xlab="Residuals")
hist(residuals(model2fit), main="Model 2 Residuals", xlab="Residuals")
```



The linear model (1) has a long left-hand tail in the distribution of residuals. Model 1 also

has several cases in the middle of its fitted-value range with large positive residuals. These correspond to points that are much closer to the left sides of the plots for the models on linear scales. They correspond to abalone with much more meat than one would expect based on their size.

There are four abalone whose `Whole.weight` is less than their `Shucked.weight`. Perhaps these should have been deleted. Both models have residuals with long upper tails.

The variance of the residuals does not seem to be constant; the variance increases with larger fitted values. So not all our regression assumptions are satisfied.

(c) Calculate the **training** error for both models. Which model fits better to the training data?

Solution: We can do this easily by simply taking the mean of the squared residuals, but we must be careful with Model 1, since it predicts $\log(Y)$ instead of Y. We'll take the difference between its predictions on the original scale and the true values.

```
m1_trainerr <- mean((exp(fitted(model1fit)) - abalone$Shucked.weight)^2)
m1_trainerr
## [1] 5393.721
m2_trainerr <- mean(residuals(model2fit)^2)
m2_trainerr
## [1] 6408.51
```

The linear model appears to fit the training data better, because it has lower squared error.

(d) Perform five-fold cross-validation to estimate the mean squared error, and compute a rough estimate of the standard errors. Based on this result and your residual analysis, which model appears to perform best?

Solution: Note that there are 4173 abalone, which is not divisible by 5 for five-fold CV. But it doesn't matter much if the folds are different in size by one or two abalone, since that's quite a small difference in relative terms.

```
n <- nrow(abalone)
thefolds <- sample(rep(1:5, length.out = n), replace = FALSE)
cv <- matrix(NA, 2, 5)
rownames(cv) <- c("Model 1", "Model 2")

## Loop through the folds, doing both models together
for (fold in 1:5) {
  ## Set up training and test data
  train <- abalone[thefolds != fold,]
  test <- abalone[thefolds == fold,]

  ## Model 1
  out <- lm(log(Shucked.weight) ~ log(Length) + log(Diameter) + log(Height),
            data=train)
  cv[1, fold] <- mean((test$Shucked.weight -
                         exp(predict(out, newdata=test)))^2)

  ## Model 2
  out <- smooth.spline(y = train$Shucked.weight,
                       x = train$Length * train$Diameter * train$Height)
  cv[2, fold] <- mean(
  (test$Shucked.weight -
     predict(out, x = test$Length * test$Diameter * test$Height)$y)^2)
}
```

Now let's look at what we got:

```
apply(cv, 1, mean)
##  Model 1  Model 2
## 5408.915 6619.652
apply(cv, 1, sd)/sqrt(5)
##  Model 1  Model 2
## 158.8183 212.1560
```

The difference in cross-validation errors between Models 1 and 2 is large compared to the naively-calculated standard errors (treating the five folds as uncorrelated).

Model 1 is the simplest model (with lowest complexity) and it is also easier to fit and interpret than Model 2. The sample distributions of the residuals from both models have similar features so based on the residuals there is no reason to choose one model over the other.

Although we are not doing a formal hypothesis test here, we can say that based on prediction performance and the residual analysis there seems to be no reason to choose a more complex model like a smoothing spline over a simpler model like linear regression.

So, Model 1 comes out on top.

(e) Under the assumption that $\hat{\beta}$ from the linear model is multivariate normal, compute a 95% confidence interval for $\beta_{\text{log-Diameter}}$. Does it include 0? Confirm your calculations with the `confint` command. Based on the EDA and residual analysis, comment on how reliable you think this confidence interval is.

Hint: You can retrieve an estimate of the variance-covariance matrix using the `vcov` command in `R`. For the question, think about what assumptions went into the computation of this confidence

interval, and whether it is reasonable to assume that $\hat{\beta}$ is multivariate normal.

Solution:

```
vbetas <- vcov(model1fit)
cibeta_logdiam <- coef(model1fit)["log(Diameter)"] +
    qnorm(0.975) * c(-1,1) * sqrt(vbetas["log(Diameter)", "log(Diameter)"])

cibeta_logdiam
## [1] 0.6933160 0.9732083
## version from confint
confint(model1fit, parm="log(Diameter)")
##                   2.5 %    97.5 %
## log(Diameter) 0.6932753 0.9732489
```

Both confidence intervals are quite similar, and neither includes 0.

Given our reasonably large sample size, the non-normality of residuals isn't too concerning here. The unequal variance, however, is, and suggests this confidence interval may not be wholly reliable, because our estimate of the variance of the residuals will not be right.