Homework 5

Advanced Methods for Data Analysis (36-402)

Due Friday February 22, 2019, at 3:00 PM

See the syllabus for general instructions regarding homework. Note that you should **always show all your work** and submit **exactly two files** (Rmd or R file as *code*, and knitted or scanned/merged pdf or html as *writeup*). Make sure that everything you submit is readable.

1. More on Optimism of the Training Error

Recall that in HW 1, we showed that

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{r}_n(X_i))^2\right] \le \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\left(Y_i^{'} - \hat{r}_n(X_i^{'})\right)^2\right],$$

where the regression estimator $\hat{r}_n(\cdot)$ has been fit on an i.i.d training sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ of size n from some distribution $F_{X,Y}$, and the m pairs $(X_1^{'}, Y_1^{'}), \ldots, (X_m^{'}, Y_m^{'})$ represent an i.i.d. test sample from the same distribution but with all observations independent from the training data.

We are now going to look closer at how large the so-called optimism in the training error is. For simplicity, consider fixed x_i's and let Y_i^* denote the value of a future observation of Y_i at covariate value x_i. Show that

$$\mathbb{E}\left[\left(Y_i - \hat{Y}_i\right)^2\right] = \mathbb{E}\left[\left(Y_i^* - \hat{Y}_i\right)^2\right] - 2\text{Cov}(\hat{Y}_i, Y_i),$$

where $\hat{Y}_i = \hat{r}_n(x_i)$. That is, the training error underestimates the prediction risk, and $2\text{Cov}(\hat{Y}_i, Y_i)$ measures the so-called optimism of the training error relative the (in-sample) prediction risk.

Hint: Follow what we discussed in lecture, and make sure you carefully explain all the steps taken in your calculations.

Solution

Let $\bar{r}(x_i) = \mathbb{E}[\hat{r}_n(x_i)]$, then

$$\begin{aligned}
\mathbb{E}\left[\left(Y_i - \hat{Y}_i\right)^2\right] &= \mathbb{E}\left[(Y_i - r(x_i) + r(x_i) - \bar{r}(x_i) + \bar{r}(x_i) - \hat{r}_n(x_i))^2\right] \\
&= \sigma^2(x_i) + \text{bias}^2 + \text{Var}(\hat{r}_n(x_i)) \\
&\quad + 2\mathbb{E}\left[(Y_i - r(x_i))(r(x_i) - \bar{r}(x_i))\right] + 2\mathbb{E}\left[(r(x_i) - \bar{r}(x_i))(\bar{r}(x_i) - \hat{r}_n(x_i))\right] \\
&\quad - 2\mathbb{E}\left[(Y_i - r(x_i))(\hat{r}(x_i) - \bar{r}_n(x_i))\right] \\
&= \sigma^2(x_i) + \text{bias}^2 + \text{Var}(\hat{r}_n(x_i)) - 2\text{Cov}(Y_i, \hat{Y}_i)
\end{aligned}$$

where the last equality follows because $r(x_i)$ and $\bar{r}(x_i)$ are constants when the x's are fixed, $\mathbb{E}[Y_i - r(x_i)] = 0$, and $\mathbb{E}[\bar{r}(x_i) - \hat{r}(x_i)] = 0$ by the definitions of $r(x_i)$ and $\bar{r}(x_i)$.

Finally, notice that

$$\begin{aligned}\mathbb{E}\left[\left(Y_i^* - \hat{Y}_i\right)^2\right] &= \mathbb{E}\left[(Y_i^* - r(x_i) + r(x_i) - \bar{r}(x_i) + \bar{r}(x_i) - \hat{r}_n(x_i))^2\right] \\ &= \sigma^2(x_i) + \text{bias}^2 + \text{Var}(\hat{r}_n(x_i))\end{aligned}$$

by the same logic as before. This time, however, even the third cross-term (the "covariance" term) cancels by independence of Y^* and $\hat{r}_n(x_i)$. In brief, as shown in lecture, this is just the expression for $MSE(x_i) + \sigma^2(x_i)$, the sum of the prediction mean-squared error and the variance of the irreducible error at $X = x_i$.

2. Housing Data (revisited)

Return to the housing data with which you worked in Homework 2. In this problem, we will take into account the locations of the census tracts using some combination of the two location variables `Latitude` and `Longitude`.

For parts (a) through (d) below, use *only* the training data set. Also, draw all scatter plots using the graphical parameter `,pch="."` in order to cut down on the overlap of points.

Part a

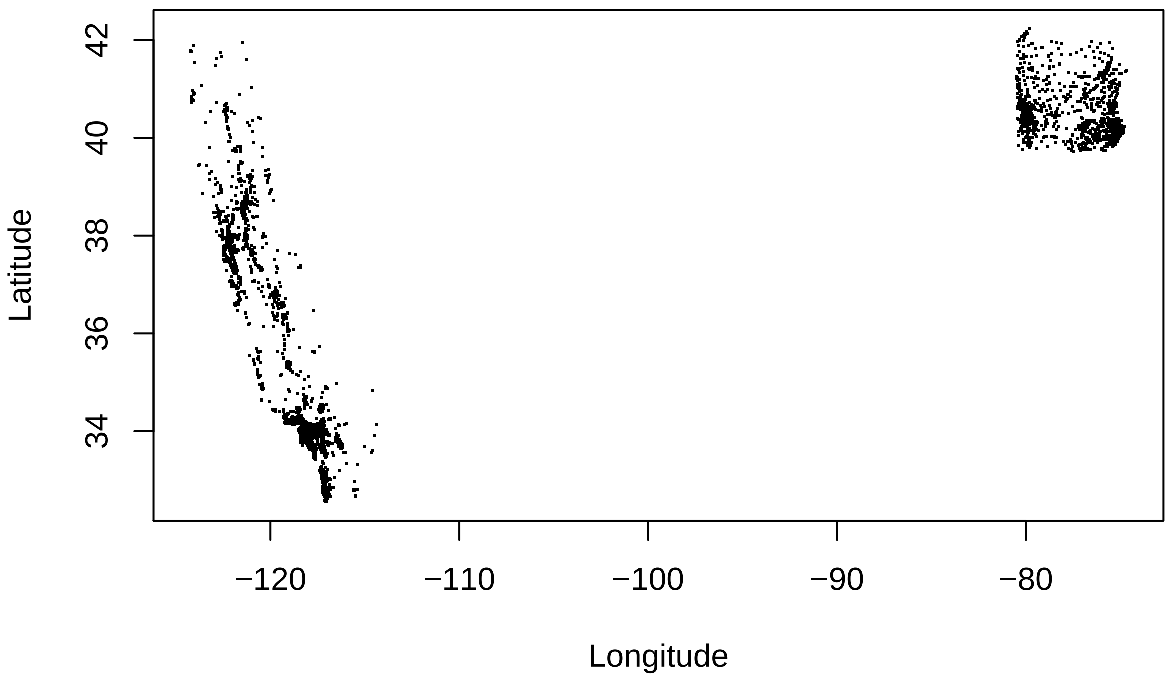
Draw the scatter plot of `Latitude` and `Longitude`. From the scatter plot, identify at least two places, locations in (`Latitude`, `Longitude`) space, where census tracts cluster.

Solution

```
housetrain=read.csv("housetrain.csv",header=T)
housetest=read.csv("housetest.csv",header=T)
names(housetrain)
```

```
## [1] "Population"              "Latitude"
## [3] "Longitude"               "Median_house_value"
## [5] "Median_household_income" "Mean_household_income"
```

```
attach(housetrain)
plot(Longitude,Latitude,pch=".")
```



The census tracts are in Pennsylvania (those with `Longitude`$>$ 80) and in California (those with `Longitude`< -115.) There are two clusters in Pennsylvania that correspond to Pittsburgh (`Longitude` close to -80 and `Latitude` between 40 and 41) and Philadelphia (highest `Longitude`> -75.5 and lowest `Latitude`.) There are also a number of clusters in California corresponding to the San Francisco, Los Angeles, and San Diego areas.

Part b

Whether you did this or not in Homework 2, plot the residuals from Model 3 (the one with both `Mean_household_income` and `Median_household_income` as predictors) against each of the location variables. Are there patterns that suggest that location might be useful in predicting house values? Explain.

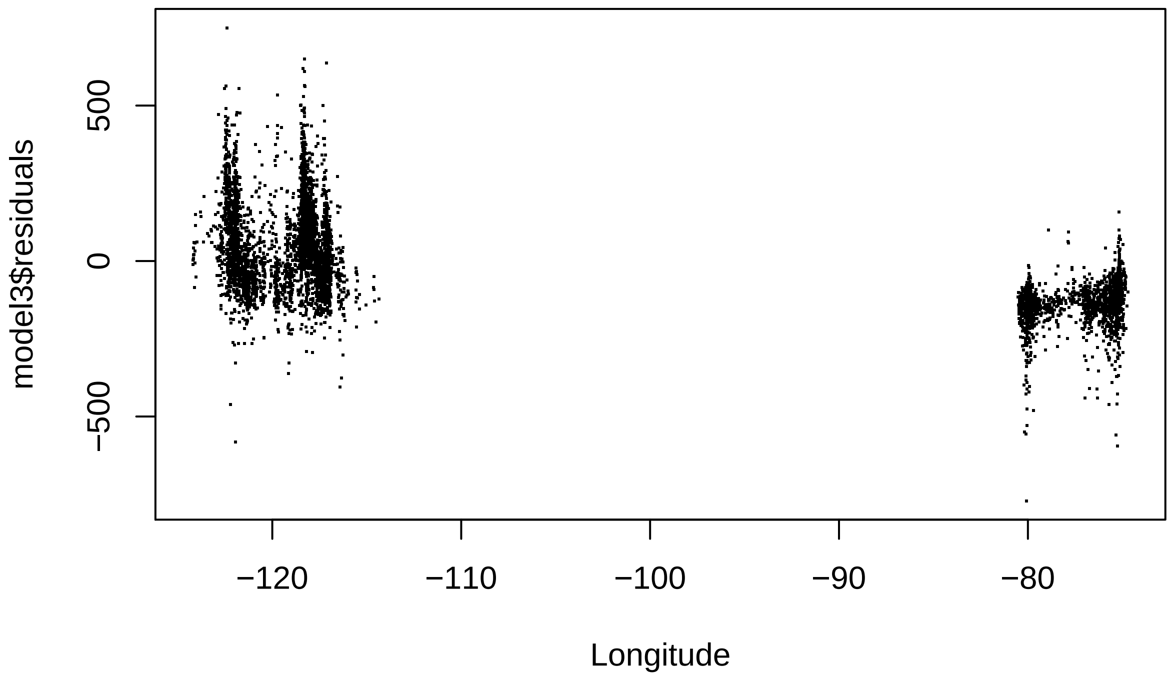
Solution

```
model3=lm(Median_house_value ~ Mean_household_income + Median_household_income)
summary(model3)
```

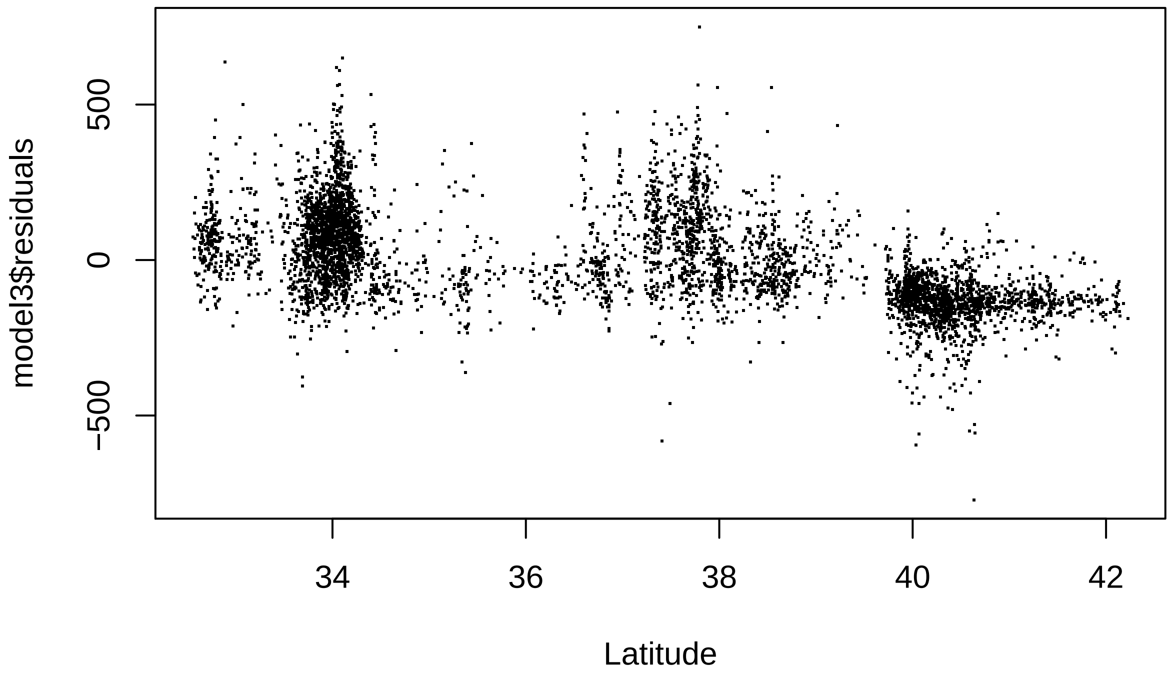
```
##
## Call:
## lm(formula = Median_house_value ~ Mean_household_income + Median_household_income)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -770.87 -112.34  -21.68   97.43  749.98
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -3.4325926  5.3880615  -0.637    0.524
## Mean_household_income    0.0054576  0.0002107  25.901  < 2e-16 ***
## Median_household_income -0.0010032  0.0002511  -3.996 6.53e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 151.3 on 5300 degrees of freedom
## Multiple R-squared:  0.4834, Adjusted R-squared:  0.4832
## F-statistic:  2480 on 2 and 5300 DF,  p-value: < 2.2e-16
```

```
plot(Longitude,model3$residuals,pch=".")
```



```
plot(Latitude,model3$residuals,pch=".")
```



The `Longitude` plot shows two striking features. First, house prices in California are higher than in Pennsylvanis (OK, not so striking) and second that prices seem more spread out in California than in Pennsylvania. These same features show up in the `Latitude` plot, where the left and central clusters are Southern and Northern California, and the right cluster is mostly Pennsylvania.

Part c

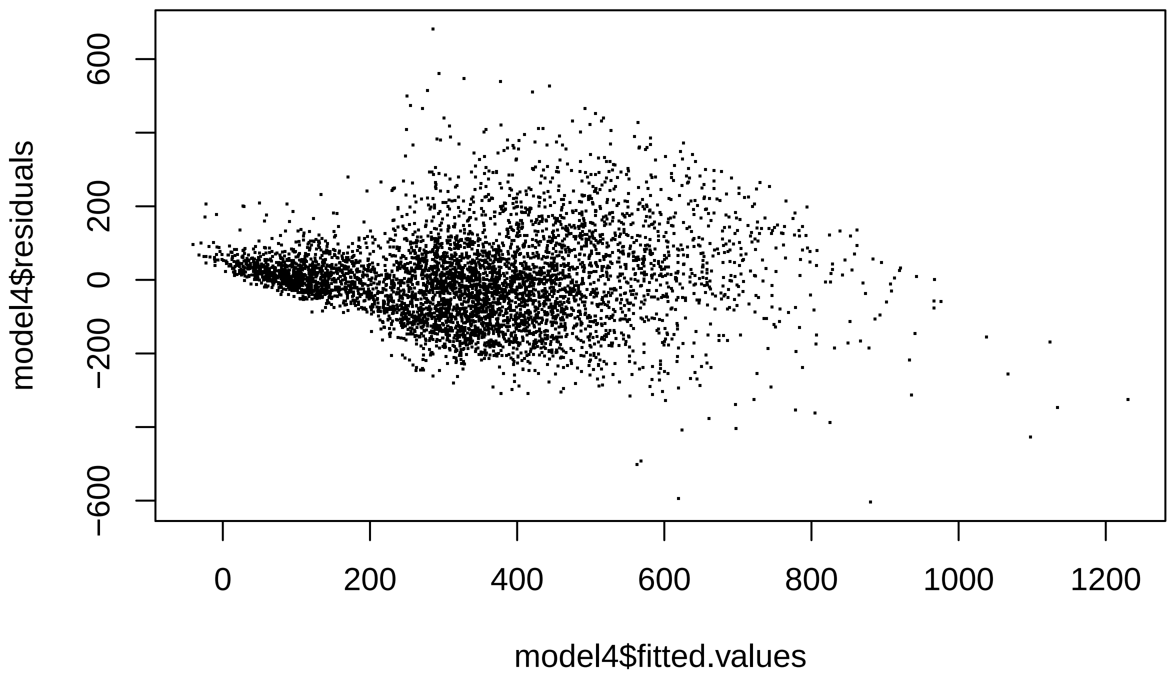
The most straightforward way to take location into account is to include the two location variables in a linear model. Define Model 4 to be the linear regression of `Median_house_value` on the four predictors consisting of the two in Model 3 and the two location variables.
Fit this model and plot its residuals against the four predictors and its fitted values. Between the residual plots and the summary of the regression, explain why it appears that one of Model 3 or Model 4 is better than the other.

Solution

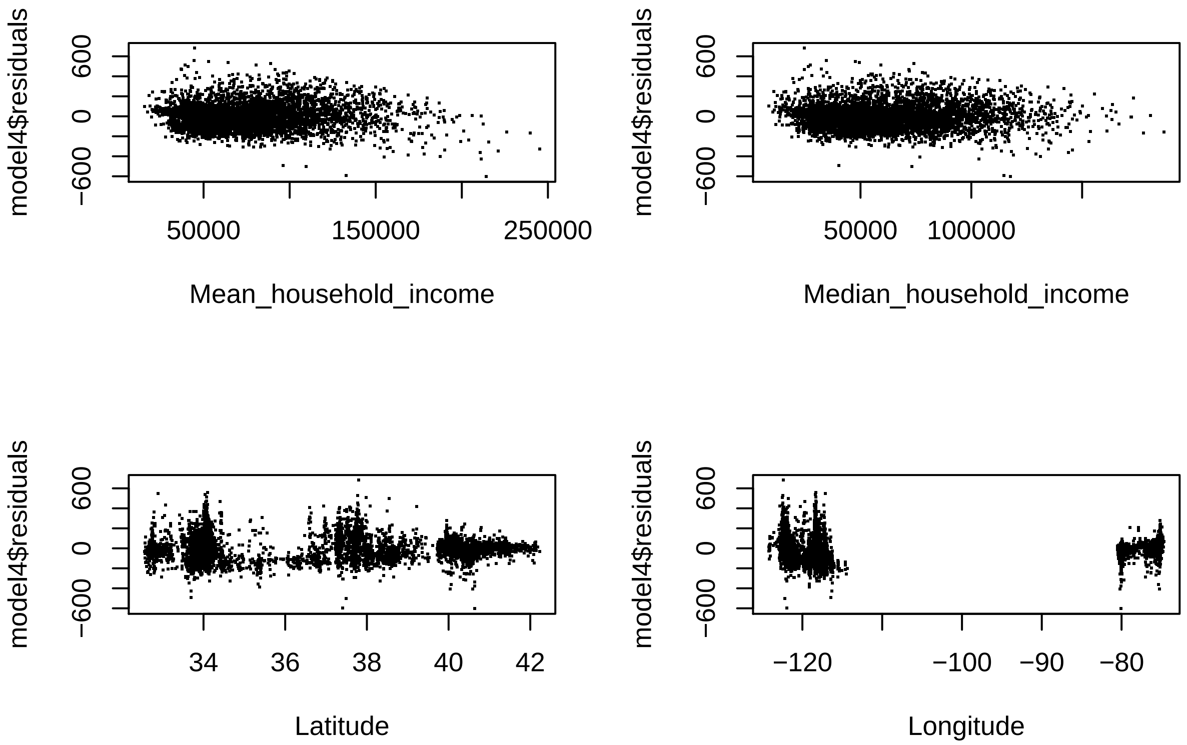
```
model4=lm(Median_house_value ~ Mean_household_income +
Median_household_income+Latitude+Longitude)
summary(model4)
```

```
##
## Call:
## lm(formula = Median_house_value ~ Mean_household_income + Median_household_income +
##     Latitude + Longitude)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -603.90  -75.39   -2.33   59.12  681.32
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              7.945e+01  4.138e+01   1.920   0.0549 .
## Mean_household_income    5.558e-03  1.659e-04  33.498   <2e-16 ***
## Median_household_income -1.761e-03  1.981e-04  -8.889   <2e-16 ***
## Latitude                -1.145e+01  8.282e-01 -13.825   <2e-16 ***
## Longitude               -3.535e+00  1.250e-01 -28.283   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 119.2 on 5298 degrees of freedom
## Multiple R-squared:  0.6798, Adjusted R-squared:  0.6796
## F-statistic:  2812 on 4 and 5298 DF,  p-value: < 2.2e-16
```

```
plot(model4$fitted.values,model4$residuals,pch=".")
```



```r
par(mfrow=c(2,2))
plot(Mean_household_income,model4$residuals,pch=".")
plot(Median_household_income,model4$residuals,pch=".")
plot(Latitude,model4$residuals,pch=".")
plot(Longitude,model4$residuals,pch=".")
```



The R^2 is a lot bigger from Model 4 compared to Model 3, and the residuals of Model~4 in the various location clusters are centered at near 0, unlike the residuals of Model 3.

Part d

It might be naive to assume that `Longitude` affects housing prices linearly over the long distance from California to Pennsylvania. For example, if house prices in Philadelphia (`Longitude` about -75 and `Latitude` about 40) are higher than house prices in Pittsburgh (`Longitude` about -80 and `Latitude` about 40.5), a linear contribution of `Longitude` that picks up the difference in price will be making a huge negative contribution to prices in California (`Longitude` about -120). But prices in California are also higher than in Pittsburgh.
To avoid such considerations, a nonparametric model in which the conditional mean of `Median_house_value` given the predictors is allowed to be a nonlinear function of the predictors. Fit two additional models:

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

Model 6: A kernel regression of `Median_house_value` on `Median_household_income`, `Mean_household_income`, `Latitude`, and `Longitude`.

Once again, you will need to load the `np` package with `library(np)`. You need a different bandwidth for each predictor. For bandwidths, use the sample standard deviations of each predictor divided by $n^{1/5}$, where n is the number of data points used to fit the model. [1]

For example, in Model 6, the predictors in the order stated are variables 5, 6, 2, and 3 in the supplied data files. So the bandwidths can be supplied as the argument

```
traindat <- read.csv("housetrain.csv", header=TRUE)
n <- nrow(traindat)
bws <- apply(traindat[, c(5,6,2,3)], 2, sd)/n^(0.2)
```

to the `npreg` command, where n is the size of the training data, (5303 in this case but *different* in part (e).) Plot residuals against the fitted values and predictors, and comment on any patterns you see. The residuals will be in the fitted object in an item called `resid` if you add the argument `residuals=T` when you call `npreg`. The fitted values will be in an item called `mean` if you *don't* specify any `newdata`.

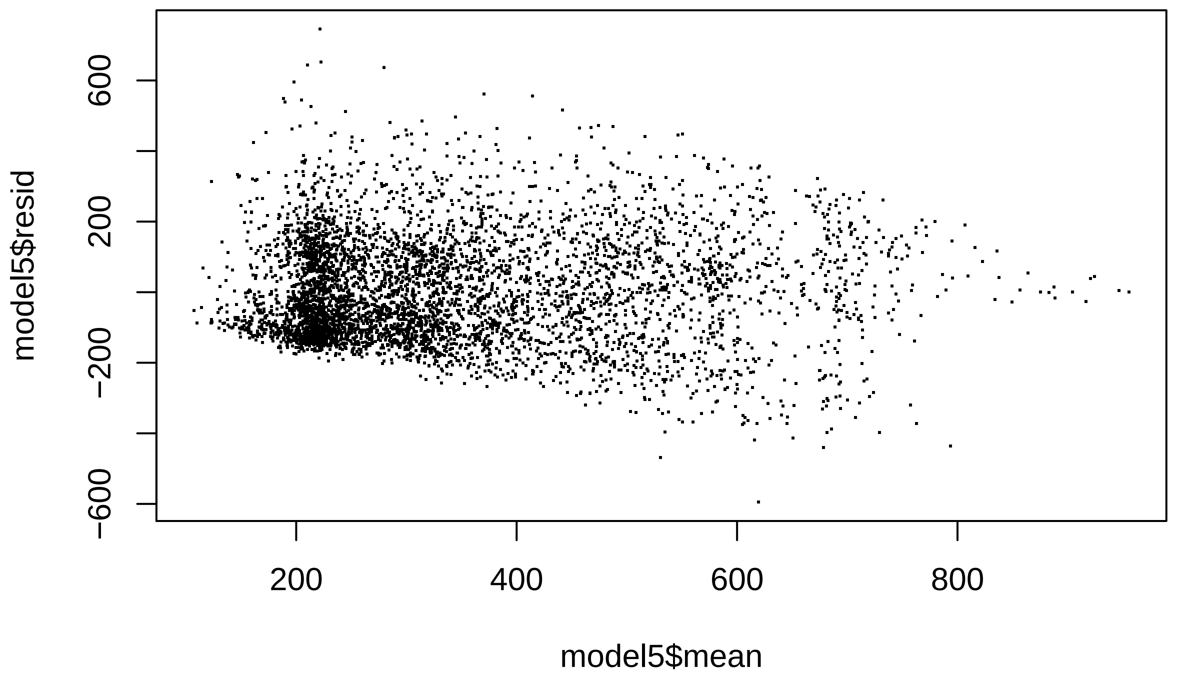
Solution

```
# Load the np library
library(np)
```

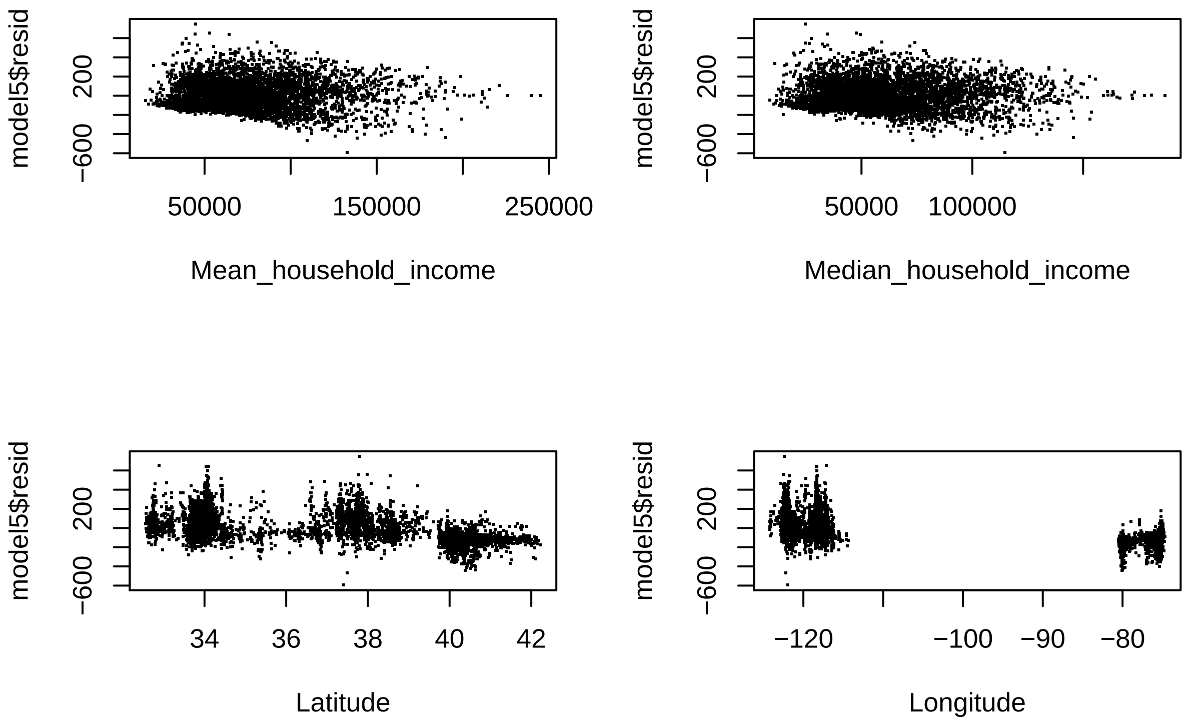
```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-9)
## [vignette("np_faq",package="np") provides answers to frequently asked questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

```
# Compute the bandwidts
bw=apply(housetrain[,c(5,6,2,3)],2,sd)/nrow(housetrain)^(0.2)
# Model 5
model5=npreg(Median_house_value~Mean_household_income+
               Median_household_income,bws=bw[c(1,2)],residuals=T)

par(mfrow=c(1,1))
plot(model5$mean,model5$resid,pch=".")
```

[1]The choices of bandwidths in these exercises was made based on some theory that will appear in a later lecture. The specific choices are not "optimal," but they are popular rule-of-thumb choices.

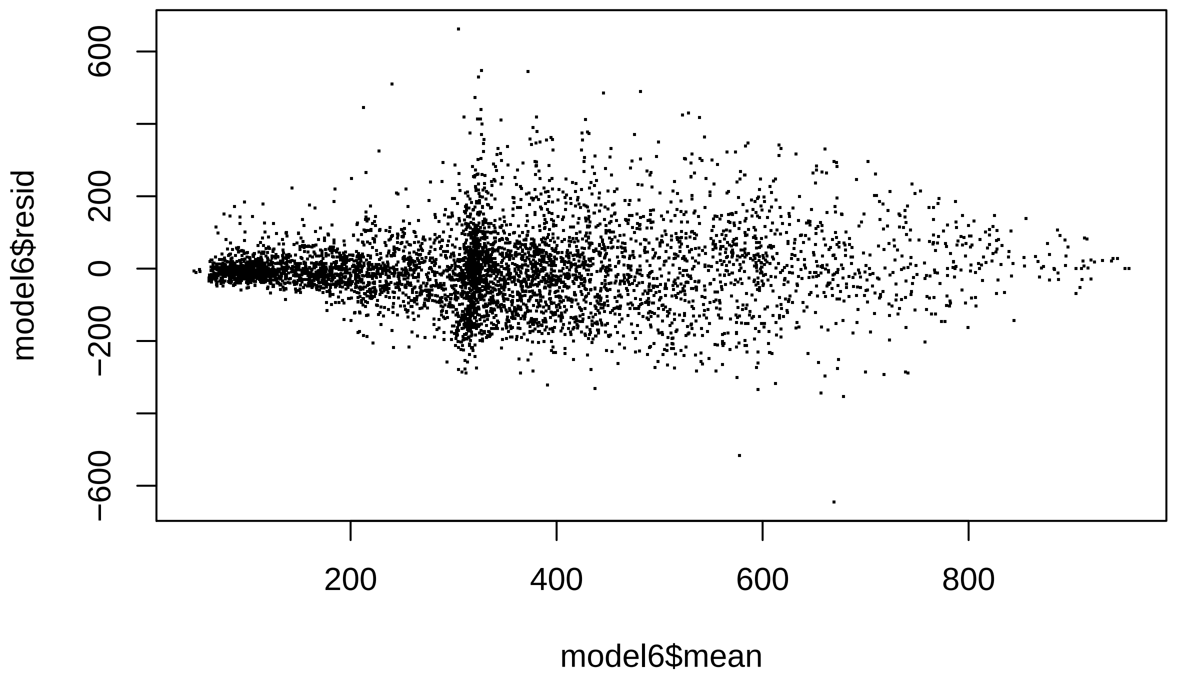


```
par(mfrow=c(2,2))
plot(Mean_household_income,model5$resid,pch=".")
plot(Median_household_income,model5$resid,pch=".")
plot(Latitude,model5$resid,pch=".")
plot(Longitude,model5$resid,pch=".")
```

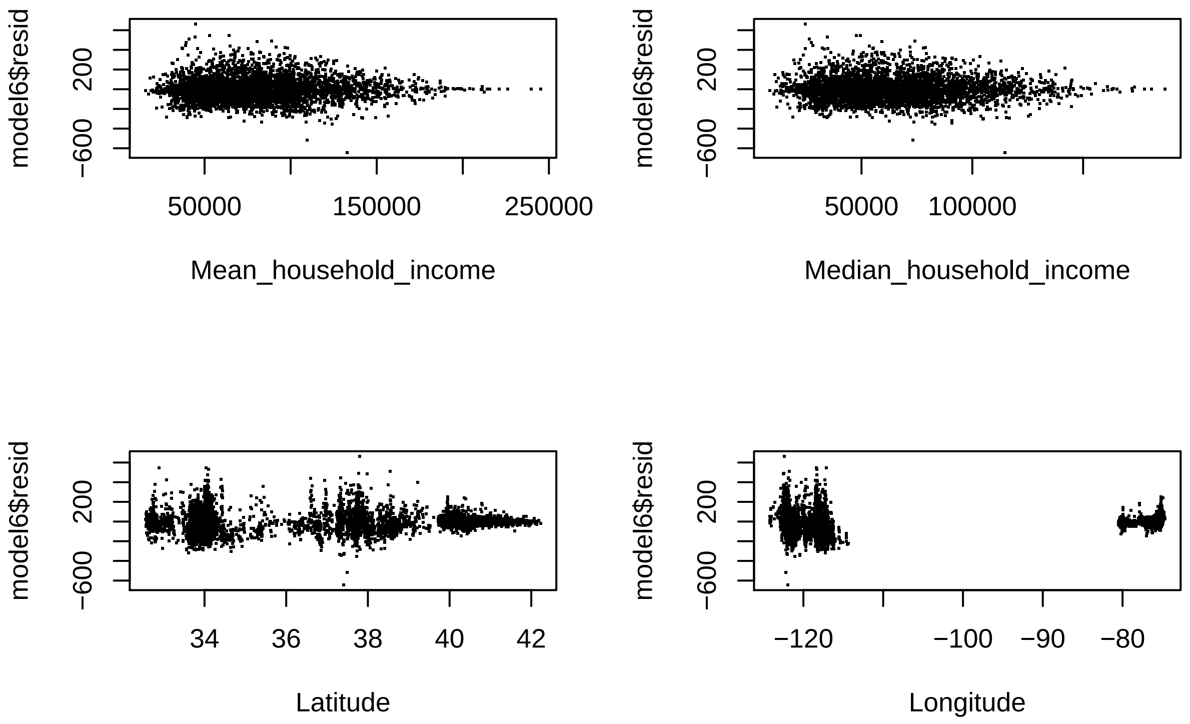


```
# Model 6
model6=npreg(Median_house_value~Mean_household_income+
               Median_household_income+Latitude+Longitude,
             bws=bw,residuals=T)
```

```
par(mfrow=c(1,1))
plot(model6$mean,model6$resid,pch=".")
```



```
par(mfrow=c(2,2))
plot(Mean_household_income,model6$resid,pch=".")
plot(Median_household_income,model6$resid,pch=".")
plot(Latitude,model6$resid,pch=".")
plot(Longitude,model6$resid,pch=".")
```



The Model 5 plots look a lot like the Model 3 plots, especially reproducing the facts that the residuals from different locations have different means and variances. The Model 6 plots look better in that the residuals from the different locations are centered closer to zero, and the plot of residuals against fitted values is the closest to a random scatter amongst all four models. The residuals from different locations still have different variances.

Part e

Now we will perform 5-fold cross-validation to choose between Models 3–6 as predictors. For each fold of cross-validation, create the test set by randomly selecting $n_k = 2121$ for each $k = 1, \ldots, 5$. The test folds and corresponding training portions can be created in a manner similar to Demo 3:

```r
dat <- read.csv("housing.csv", header=TRUE)
nfold <- 5
samp <- sample(rep(1:nfold, ceiling(n/nfold))[1:n])

for(k in 1:nfold){
 testd <- dat[which(samp==k), ]
 traind <- dat[-which(samp==k), ]
 ## Do stuff ##
}
```

The `npreg` function will compute predictions while fitting the model if you run it with a command like

```r
library(np)
model5 <- npreg(Median_house_value ~ Median_household_income +
                Mean_household_income, data=traind, newdata=testd,
                bws=apply(traind[, c(5,6)], 2, sd)/nrow(traind)^(0.2))
```

This time n is the size of the `traind` data set (8484), which is different from the n in part (d). The predictions will then be in `model5$mean`. For each fold k and each model $m = 3, 4, 5, 6$, report the resulting values of the average squared prediction error within fold k for Model m. You can write a loop to do all the calculations and store the results in a 5×4 matrix.

Solution

```r
housedata=read.csv("housing.csv", header=TRUE)
housedata$Median_house_value <- housedata$Median_house_value/1000
samp=sample(rep(1:5,2121),replace=FALSE)
prederr=matrix(NA,5,4)
```

Next, we need to loop through the five folds, fitting the four models once with each fold, and computing the sum of squares of the prediction errors. For each fold, we need to recompute the bandwidths for the kernel regressions using the new training data.

```r
for(k in 1:5){
   testd=housedata[samp==k,]
   traind=housedata[!(samp==k),]
   model=lm(Median_house_value ~ Mean_household_income + Median_household_income,data=traind)
   prederr[k,1]=mean((predict(model,newdata=testd)-testd$Median_house_value)^2)
   model=lm(Median_house_value ~ Mean_household_income + Median_household_income+
              Latitude+Longitude,data=traind)
   prederr[k,2]=mean((predict(model,newdata=testd)-testd$Median_house_value)^2)
   bw=apply(traind[,c(5,6,2,3)],2,sd)/8484^(0.2)
   model=npreg(Median_house_value~Median_household_income+
                 Mean_household_income,data=traind,newdata=testd,
                 bws=bw[c(1,2)])
   prederr[k,3]=mean((model$mean-testd$Median_house_value)^2)
   model=npreg(Median_house_value~Median_household_income+
                  Mean_household_income+Latitude+Longitude,
                  data=traind,newdata=testd,bws=bw)
   prederr[k,4]=mean((model$mean-testd$Median_house_value)^2)
}
```

All of the action is in the summary statistics that we compute for part (f). As a note, one could have used the `predict` function to compute the predictions from models 5 and 6 instead of including the parameter `newdata=testd` in the `npreg` function.
To use `predict`, one would have used
`mean((predict(model,newdata=testd)-testd$Median_house_value)`^{2}`)`
to compute the `prederr` values for models 5 and 6.

Part f

Compute the average of the five cross-validation error values for each model along with the corresponding standard error as defined in lecture. Comment on which model or models are clearly better than the others. How much better is the best model than the second best compared to the standard errors? Comment on what this says about how good the models are compared to each other.

Solution

```
# Report the prediction errors
prederr
```

```
##          [,1]     [,2]     [,3]     [,4]
## [1,] 24245.52 14986.13 23724.63 12731.41
## [2,] 23104.10 14807.92 22649.55 12067.12
## [3,] 22243.97 13015.42 21970.63 11281.59
## [4,] 22912.04 14817.20 22585.97 12774.28
## [5,] 21801.40 13360.95 21413.95 11769.14
```

```
# Compute overall cross-validation error and standard errors
apply(prederr,2,mean)
```

```
## [1] 22861.41 14197.52 22468.95 12124.71
```

```
apply(prederr,2,sd)/sqrt(5)
```

```
## [1] 417.2572 416.8747 386.2887 285.5365
```

We see that the two models (4 and 6) that make use of location have much smaller cross-validation error than the other two, many times both of the standard errors. The other two models are closer, but Model 6 has cross-validation error that is about 2000 smaller than Model 4, while the standard errors are only a few hundred each. It looks like Model 6 predicts better than the others and its residual plots are better as well. There still seem to be different variances for different locations. Perhaps a weighted kernel regression fit could deal with that.

3. Practicing Inference

You can find the dataset on abalones on the HW 5 page. The data set 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

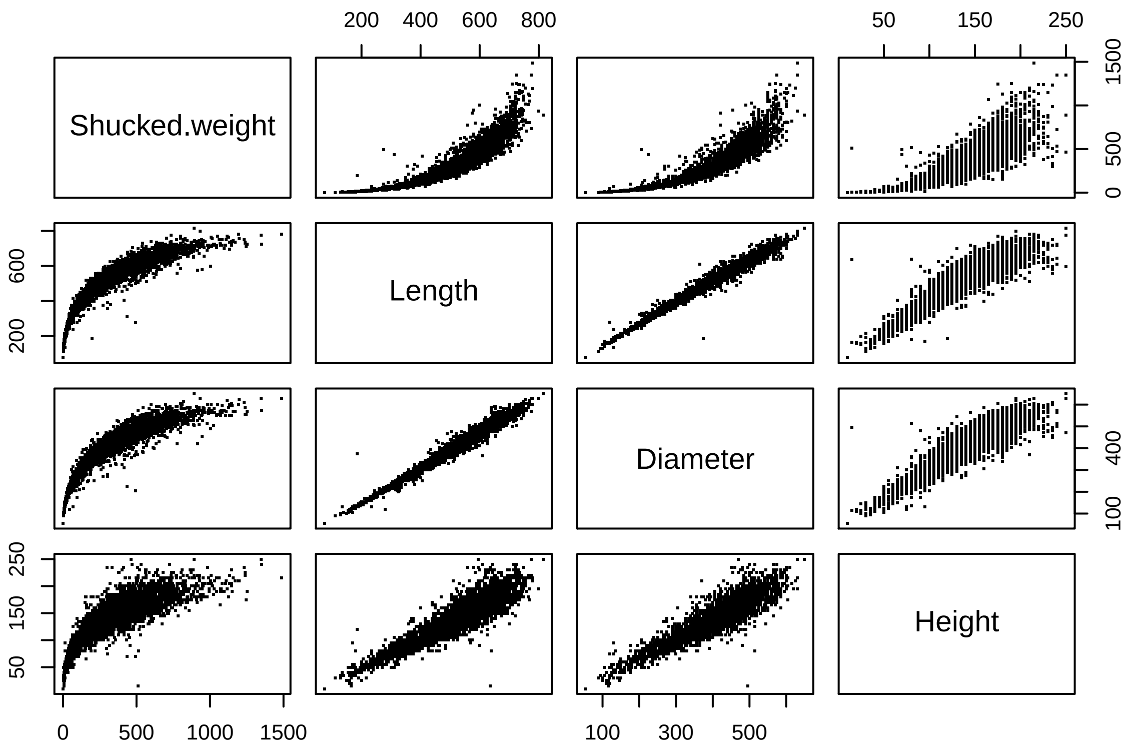
A fisherperson who wishes to sell the abalone for food is interested in the edible part. Prior to cutting one open for cooking, one can predict the weight of the edible part (`Shucked.weight`) through a regression model. A natural predictor that would be available, if a scale were available, is `Whole.weight`.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

Part a

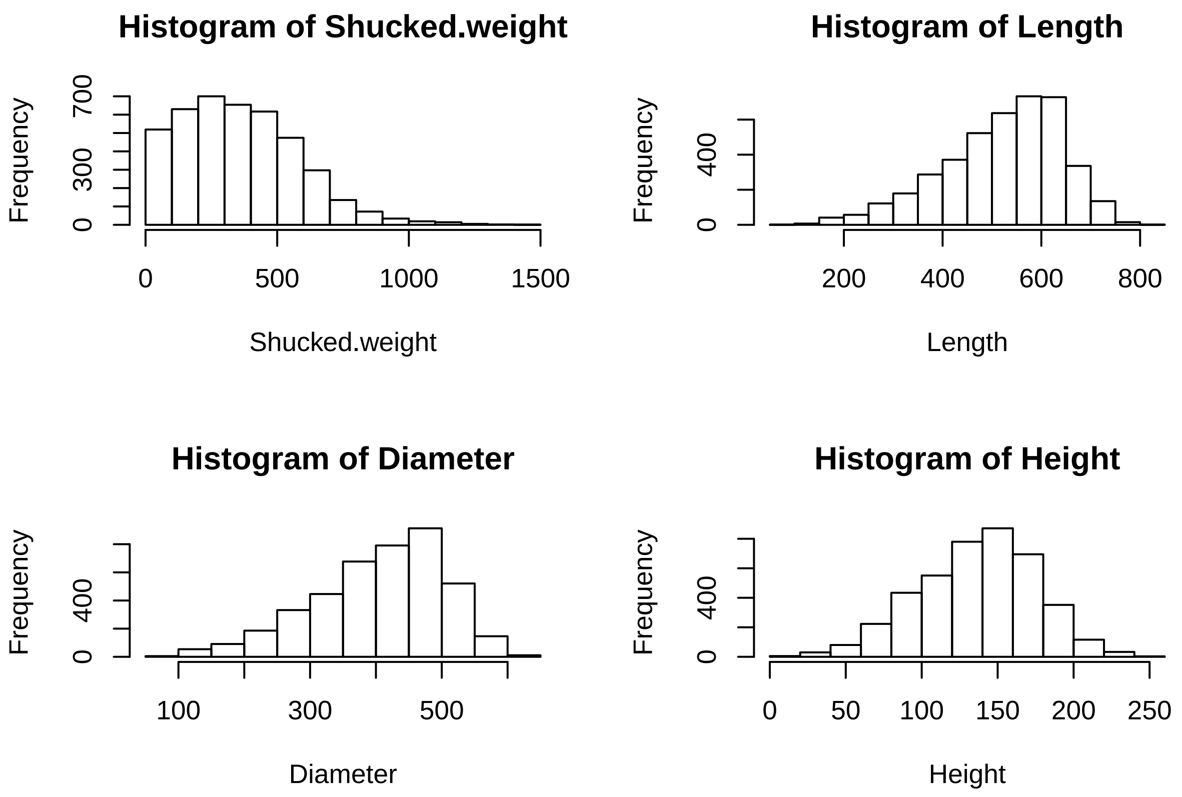
Do exploratory analysis of the response, `Shucked.weight`, and the three predictors: `Diameter`, `Length`, and `Height`. Summarize what you see from the exploratory analysis.

Solution

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

pairs(abalone[,c(6,2:4)],pch=".")
```



```r
par(mfrow=c(2,2))
hist(abalone$Shucked.weight,xlab="Shucked.weight",
main="Histogram of Shucked.weight")
hist(abalone$Length,xlab="Length",main="Histogram of Length")
hist(abalone$Diameter,xlab="Diameter",main="Histogram of Diameter")
hist(abalone$Height,xlab="Height",main="Histogram of Height")
```

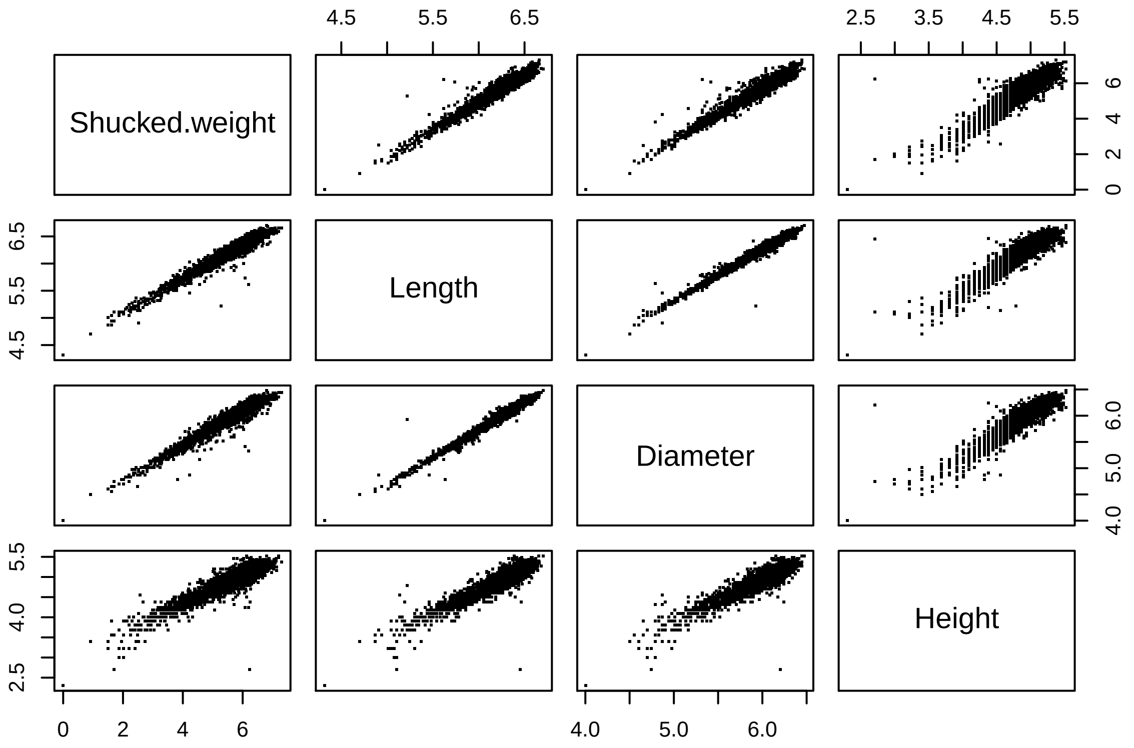


```
par(mfrow=c(1,1))
```

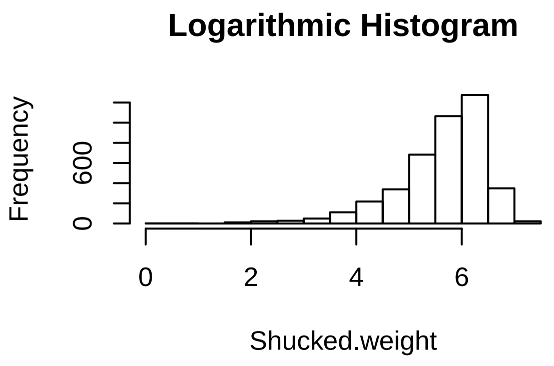
`Shucked.weight` has a strong curved relationship with the three dimension variables, which themsleves are rather linearly related. The variables are all rather skewed with the weights skewed to the right and the dimension measurements skewed to the left.

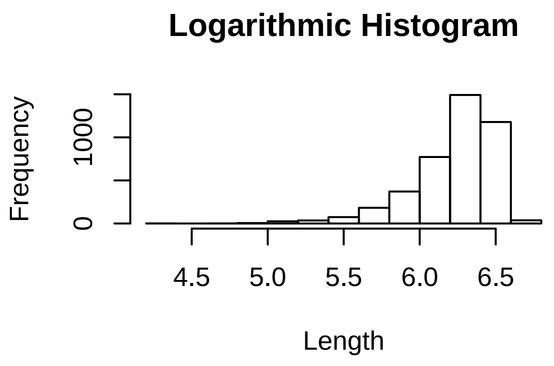
In light of the relationship between logarithms and products, we repeated the scatter plot matrix with logarithms of all four variables. Height seems to be less highly correlated with the other three variables than each of the other three are with each other. The skew of each variable seems worse in the logarithmic scale.

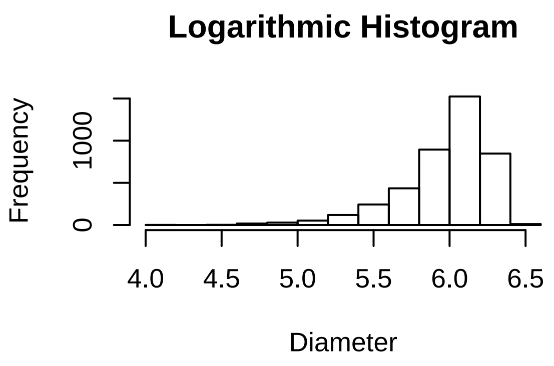
```
pairs(log(abalone[,c(6,2:4)]),pch=".")
```

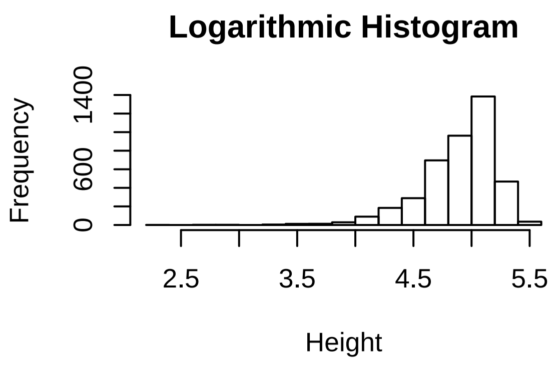


```
par(mfrow=c(2,2))

hist(log(abalone$Shucked.weight),xlab="Shucked.weight",
main="Logarithmic Histogram")
hist(log(abalone$Length),xlab="Length",main="Logarithmic Histogram")
hist(log(abalone$Diameter),xlab="Diameter",main="Logarithmic Histogram")
hist(log(abalone$Height),xlab="Height",main="Logarithmic Histogram")
```









```
par(mfrow=c(1,1))
```

Part b

We will explore linear and kernel regression models for predicting the response from some or all of the predictors. Whenever you need a bandwidth for a variable in a kernel regression, use the sample standard deviation of the variable divided by $n^{1/5}$, where n is the size of the sample. This applies to the full data or to a cross-validation calculation.

The models that you fit should include the following, but feel free to fit other models if you can think of reasons to do so:

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

Model 2: A kernel regression of `Shucked.weight`, on all three predictors: `Diameter`, `Length`, and `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.

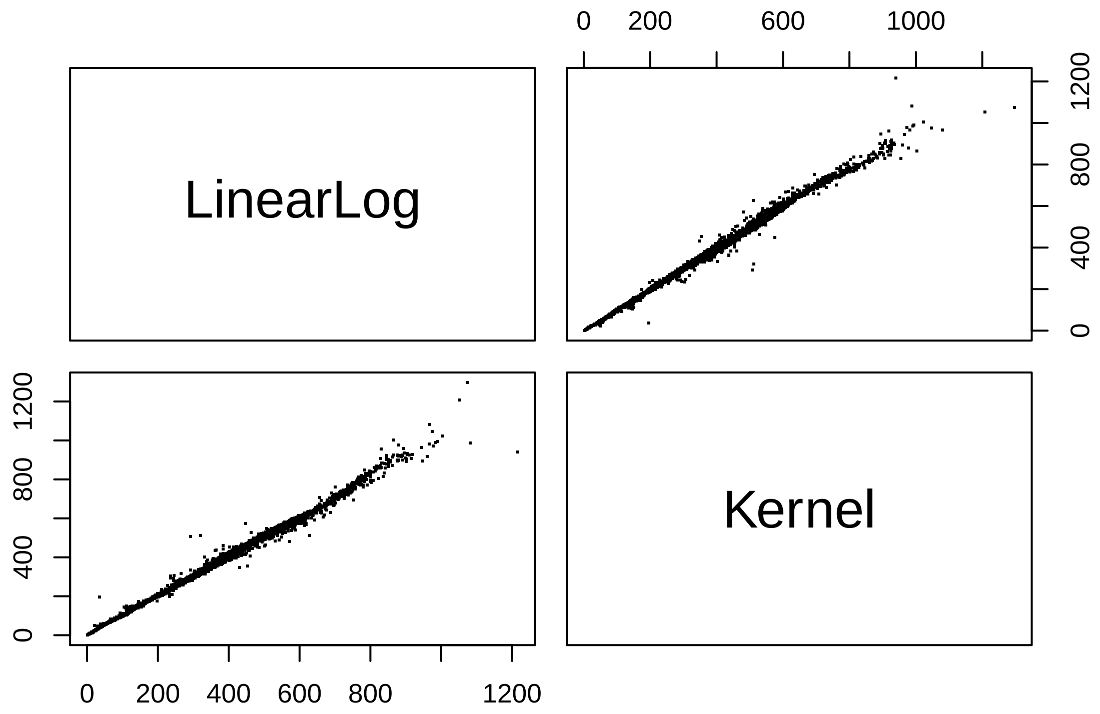
Draw a scatter plot of the fitted values for both models against each other, and comment on how similar or different the predictions seem to be.

Examine residuals from each model and say what you learn about the possible distributions of the noise terms and how those distributions might be related to other variables.

Solution

```
# Model 1: Linear regression in the log scale

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

```
# Model 2:  Kernel regression in the linear scale

library(np)
model2fit=npreg(Shucked.weight~Length+Diameter+Height,data=abalone,
bws=apply(abalone[,2:4],2,sd)/nrow(abalone)^0.2)

thefits=data.frame(LinearLog=exp(fitted.values(model1fit)),
                   Kernel=fitted.values(model2fit))

pairs(thefits,pch=".")
```



```
diag(var(thefits))
```

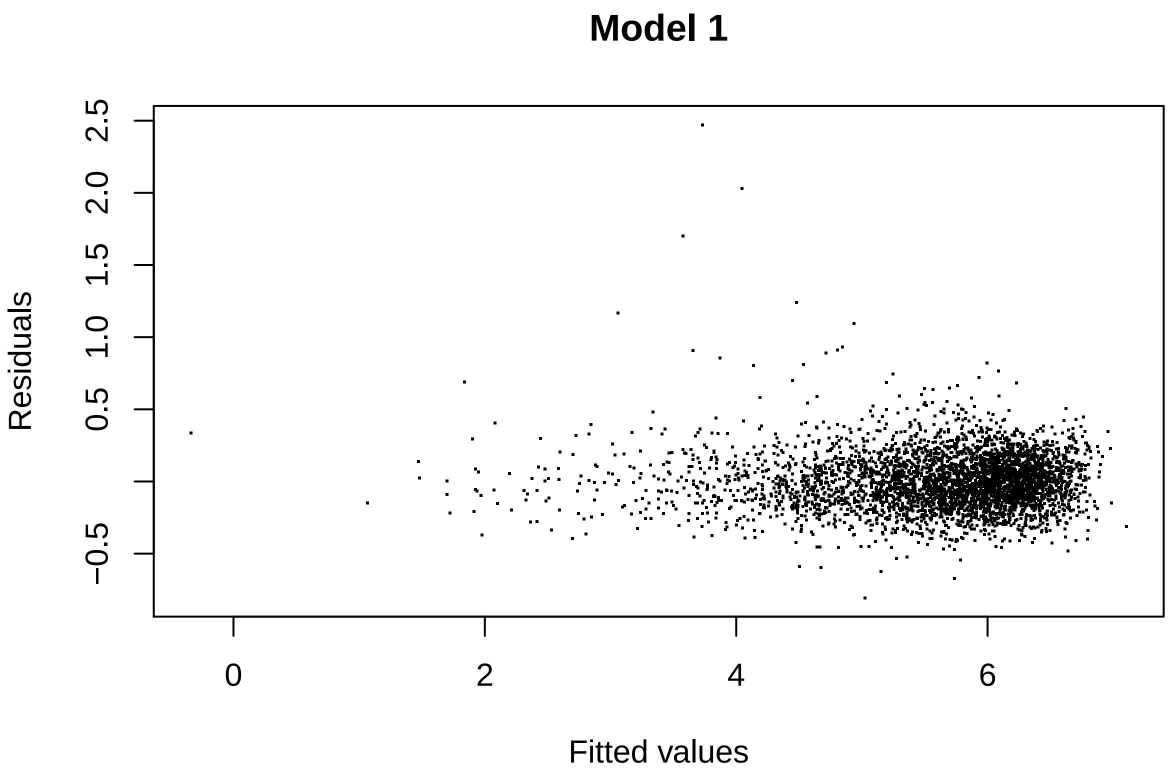
```
## LinearLog    Kernel
##  41436.27  42381.57
```

```
cor(thefits)
```

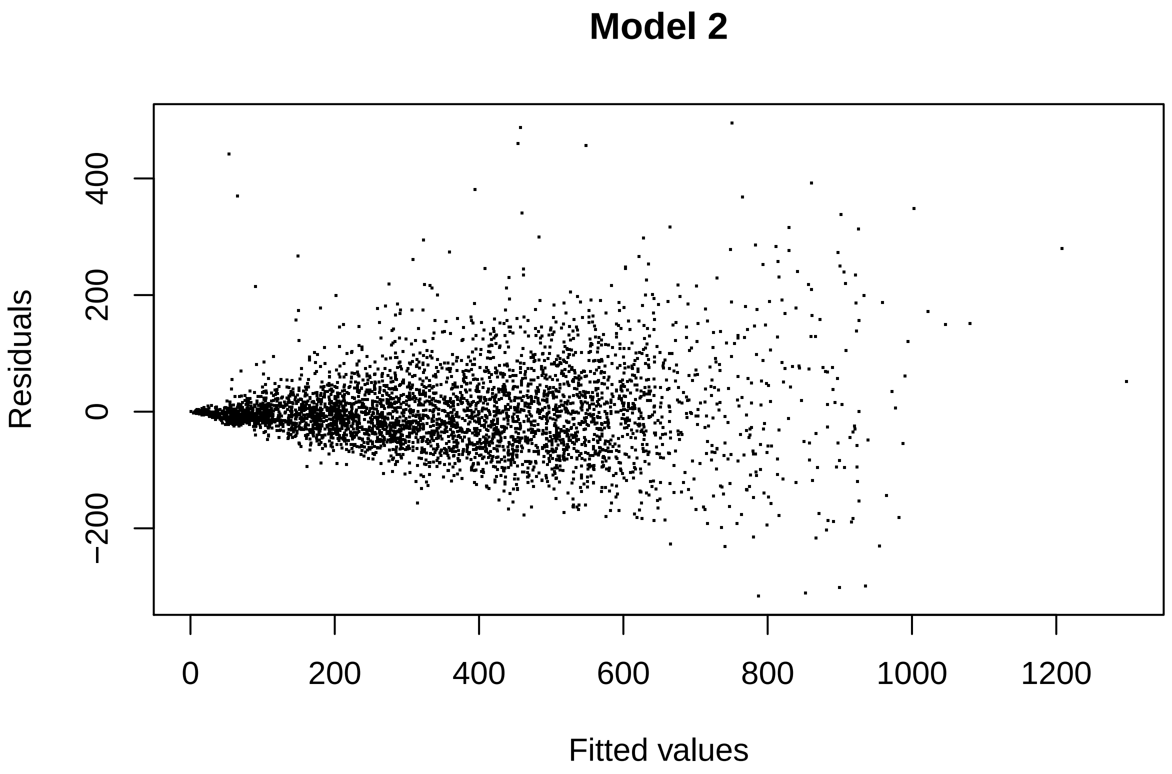
```
##           LinearLog    Kernel
## LinearLog 1.0000000 0.9974037
## Kernel    0.9974037 1.0000000
```

All of the models have highly correlated fits.

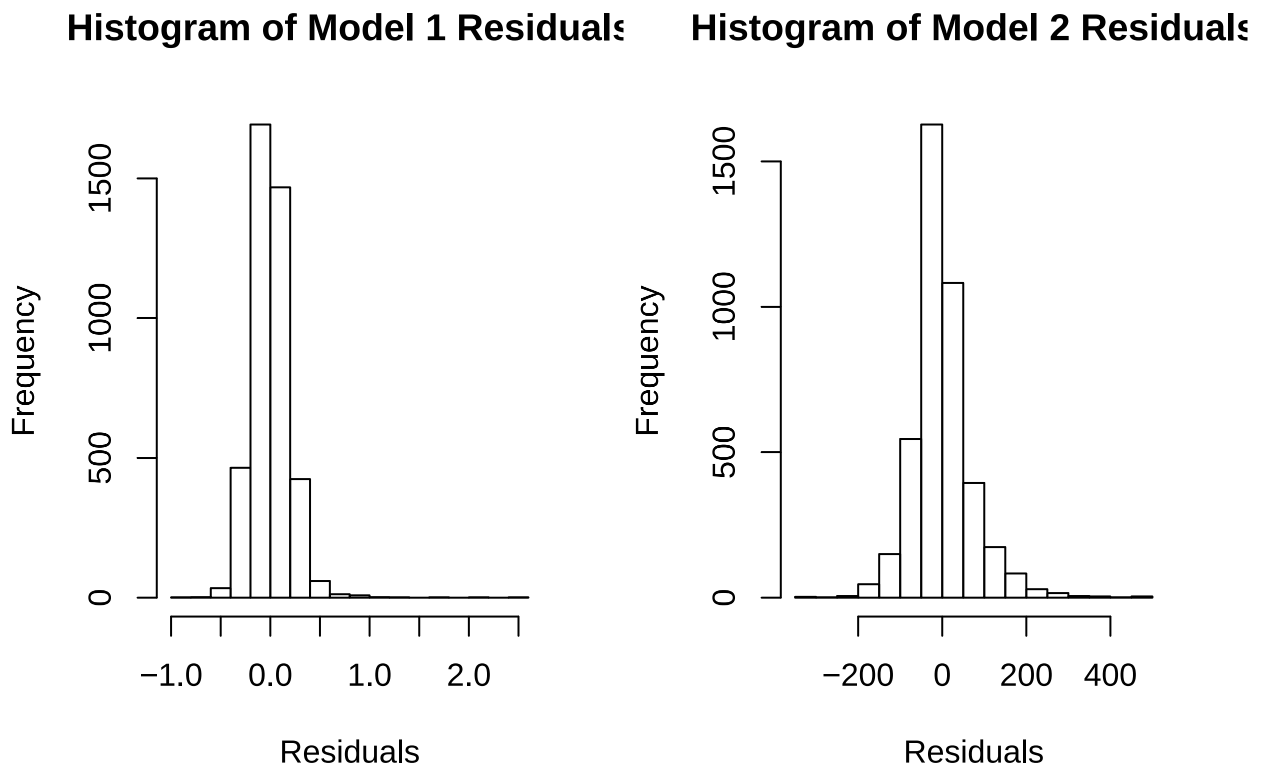
```
plot(fitted.values(model1fit),residuals(model1fit),pch=".",xlab="Fitted values",
ylab="Residuals",main="Model 1")
```



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



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



```
par(mfrow=c(1,1))
```

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.

Part c

Perform five-fold cross-validation and compute a rough estimate of its standard error. Based on this result and your residual analysis, which model appears to perform best?

```
n=nrow(abalone)
thefolds=sample(rep(1:5,length=n),replace=F)
# Loop through the folds, doing all five models together
cv=matrix(NA,3,5)
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$Shucked.weight-
        exp(predict(out,newdata=test)))^2)
# Model 2
    out=npreg(Shucked.weight~Length+Diameter+Height,data=train,
        bws=apply(train[,2:4],2,sd)/nrow(train)^0.2)
    cv[2,fold]=mean((test$Shucked.weight-predict(out,newdata=test))^2)

    out2 <- npreg(Shucked.weight~Length+Diameter+Height,data=train, newdata=test,
        bws=apply(train[,2:4],2,sd)/nrow(train)^0.2)

    cv[3,fold]=mean((test$Shucked.weight-out2$mean)^2)

}
cv
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 6200.500 5712.958 5298.492 5117.952 4704.408
## [2,] 6328.889 5855.170 5358.288 5125.153 4868.099
## [3,] 6328.889 5855.170 5358.288 5125.153 4868.099
```

```
apply(cv,1,mean)
```

```
## [1] 5406.862 5507.120 5507.120
```

```
apply(cv,1,sd)/sqrt(5)
```

```
## [1] 256.1456 261.9963 261.9963
```

Models 1 has smaller cross-validation errors than Model 2. The difference between Models 1 and 2 is not large relative to the naively-calculated standard errors (treating the five folds as uncorrelated.) You can see that the differences between the folds are not very small relative to some of the differences between the models.

The cross-validation error is smallest for Model 1 and it is the easiest to fit and use for prediction. The sample distributions of the residuals from both models have similar features, so there isn't much to choose between them on that score. So, Model 1 seems to come out one top.

Part d

Under the assumption that $\hat{\beta}$ from the linear model is jointly 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, comment on how much reliable do you think this confidence interval is.

Hint: You can retrieve an estimate of the variance-covariance matrix using the `vcov` command in `R`.

Solution

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

```
## [1] 0.6933160 0.9732083
```

```
abs(max(cibeta_logdiam - confint.default(model1fit, parm="log(Diameter)")))
```

```
## [1] 0
```

Part f

Now let's get a confidence interval for $\mathbb{E}\left\{\log(Y) \mid \text{log-Diameter} = 5.93, \text{log-Length} = 6.10, \text{log-Height} = 6.24\right\}$. Using again the variance-covariance matrix for $\hat{\beta}$, provide a 95% confidence interval under the assumption $\hat{\beta}$ is jointly normally distributed. Confirm your calculations with the `predict` function and `interval = "confidence"` argument.

Solution

```
xvals <- c(1, 6.10, 5.93, 6.24)
mean_est <- as.vector(coef(model1fit)%*%xvals)
var_mean_est <- as.vector(t(xvals)%*%vbetas%*%xvals)
ci_mean_est <- mean_est + qnorm(0.975)*c(-1,1)*sqrt(var_mean_est)
ci2 <- predict.lm(model1fit, newdata = data.frame(Length = exp(6.10),
                                                  Diameter = exp(5.93),
                                                  Height = exp(6.24)),
                  interval = "confidence")
print(ci_mean_est)
```

```
## [1] 5.555710 5.692221
```

```
## Alternatively
ci_mean_est2 <- mean_est + qt(0.975, nrow(abalone) - 4)*c(-1,1)*sqrt(var_mean_est)
abs(max(ci_mean_est2 - ci2[, c("lwr", "upr")]))
```

```
## [1] 0
```

Part g

One issue with transforming the response in a regression is the following. We defined $r(x)$ to be $\mathbb{E}\left\{Y \mid X = x\right\}$, the conditional mean of Y given $X = x$. In Model 1, we model

$$\log(Y) = \tilde{r}(x) + \varepsilon, \tag{1}$$

where ε is independent of X and has a distribution centered at 0. However, if the mean of ε is 0, $\exp\left(\tilde{r}(x)\right)$ is *not* the conditional mean of Y given $X = x$. We can still use $\exp\left(\widehat{\tilde{r}(x)}\right)$ as an estimator of $r(x)$, but it will be biased. However, let's ignore this issue for now (in practice though, this cannot be ignored. For instance, one could fit a glm instead). What is the variance of $\exp\left(\widehat{\tilde{r}(x_0)}\right)$, where $\hat{r}(x_0) = \hat{\mathbb{E}}\left[\log(Y) \mid \text{log-Diameter} = 5.93, \text{log-Length} = 6.10, \text{log-Height} = 6.24\right]$?

This is not very easy to compute, but we can be strategic. Soon, we will cover the bootstrap as a way to quantify the uncertainty when it is hard to do analytically. Here, we propose a simple trick. Do a first-order Taylor expansion of the function $f(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3) \equiv \exp\left(\widehat{\tilde{r}(x)}\right)$ around the true values of $\beta = (\beta_0, \beta_1, \beta_2, \beta_3)$. This should give you a linear expression in $\hat{\beta}$. The approximation will be good if the $\hat{\beta}$ and β are close. At this point, computing the variance is easy again and can be done with the usual estimate of the variance-covariance matrix of $\hat{\beta}$ provided by the `vcov` command.

Hint: Recall, that a first-order Taylor expansion of a function $f(\mathbf{x})$ around a point $\mathbf{a}$ is

$$f(\mathbf{x}) \approx f(\mathbf{a}) + Df(\mathbf{a})(\mathbf{x} - \mathbf{a})$$

where $Df(\mathbf{a})$ is the matrix of partial derivatives evaluated at $\mathbf{a}$ (in this case it is a 1×4 vector).

Solution

Let $\vec{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$ and $\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{bmatrix}$ Recall that a first-order Taylor expansion means that we are approximating the function in a neighborhood of β as

$$\begin{aligned} f(\hat{\beta}) &= e^{\hat{r}(\vec{x})} \\ &= \exp\left(\hat{\beta}^T \vec{x}\right) \\ &\approx \exp\left(\beta^T \vec{x}\right) + \underbrace{\begin{bmatrix} \exp\left(\beta^T \vec{x}\right) & \exp\left(\beta^T \vec{x}\right) x_1 & \exp\left(\beta^T \vec{x}\right) x_2 & \exp\left(\beta^T \vec{x}\right) x_3 \end{bmatrix}}_{\frac{\partial}{\partial \hat{\beta}} f(\hat{\beta})^T} \left(\hat{\beta} - \beta\right) \end{aligned}$$

Now that we approximated the function $f(\hat{\beta})$ by a function that is linear in $\hat{\beta}$, we can compute the variance more easily as

$$\mathrm{Var}\left(f(\hat{\beta})\right) \approx \begin{bmatrix} \exp\left(\beta^T \vec{x}\right) \\ \exp\left(\beta^T \vec{x}\right) x_1 \\ \exp\left(\beta^T \vec{x}\right) x_2 \\ \exp\left(\beta^T \vec{x}\right) x_3 \end{bmatrix}^T \mathrm{Var}\left(\hat{\beta} - \beta\right) \begin{bmatrix} \exp\left(\beta^T \vec{x}\right) \\ \exp\left(\beta^T \vec{x}\right) x_1 \\ \exp\left(\beta^T \vec{x}\right) x_2 \\ \exp\left(\beta^T \vec{x}\right) x_3 \end{bmatrix}$$

We note that $\mathrm{Var}(\hat{\beta} - \beta) = \mathrm{Var}(\hat{\beta})$, since β is just a constant. A way to estimate the quantity above is to replace β with $\hat{\beta}$, as we do in the code below:

```r
first_derivatives_est <- exp(mean_est)*xvals
se_taylor <- as.vector(sqrt(t(first_derivatives_est)%*%vbetas%*%first_derivatives_est))
ci_taylor <- exp(mean_est) + qnorm(0.975)*c(-1,1)*se_taylor
print(ci_taylor)
```

```
## [1] 258.0799 295.8914
```