Homework 7

Advanced Methods for Data Analysis (36-402)

Due Friday April 1, 2022 at **3:00 pm ET**

Note: Several calculations in this assignment are time-consuming and could take more than 5 minutes. Use the `cache=TRUE` chunk option in R Markdown so they are not re-run every time you knit. This will speed up the "Knit" button after the first time you knit, but running the code blocks manually will still be slow.

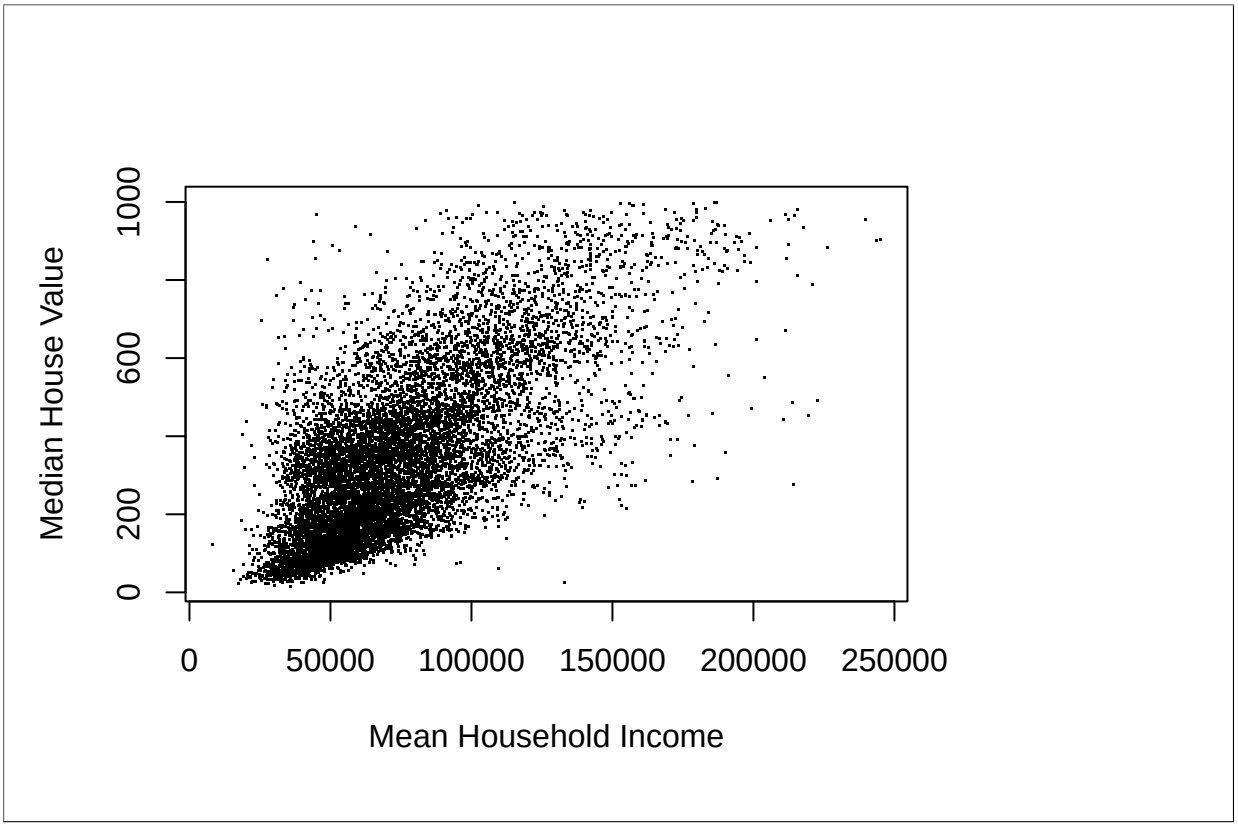
1. **Degrees of Freedom for Kernels and Splines.** Once more, use the housing data from Homeworks 2, 4, and 5. Merge the training and test data sets into a single data set as in problem 1 (b) of Homework 5.

   We will be fitting models in which we try to predict median house value from mean household income. We will use both splines and normal-kernel regressions with a variety of tuning parameters while we attempt to use "effective degrees-of-freedom" to make the comparisons on a common scale.

   (a) Plot `Median_house_value` versus `Mean_household_income`.

   Solution:

```
suppressPackageStartupMessages(library(np))

htest <- read.csv("housetest.csv", header=TRUE)
htrain <- read.csv("housetrain.csv", header=TRUE)
housing <- rbind(htrain, htest)

plot(housing$Mean_household_income, housing$Median_house_value, pch=".",
     xlab="Mean Household Income", ylab="Median House Value")
```



(b) First, fit several spline modes with different smoothing parameters. We will use `smooth.spline` with the `df` version of the smoothing parameter. (This is the same as the *effective degrees-of-freedom* discussed in Lecture 6. R will automatically select the penalty λ to match the number of effective degrees of freedom you select.)

Use `df` parameter values from 2 to 27 in steps of 1. For each `df`, run five-fold cross-validation on the full data set, computing the estimated MSE of prediction on each held-out fold. Compute the sample average of the five estimated MSE of prediction values for each `df`.

Also, for each `df`, compute an estimated standard error for the average by computing the sample standard deviation of the five estimated MSE of prediction values divided by $\sqrt{5}$. Plot the sample average of the five estimated MSE of prediction values against `df`, and find the `df` value that provides the lowest average. Compare the differences between the plotted averages to the estimated standard errors of the averages. Say why this comparison either does or does not inspire confidence that we have really found the `df` that provides the best out-of-sample prediction.

Solution: First, we initialize the `df` parameter values and create the folds. These same folds will be used again for the kernel regressions.

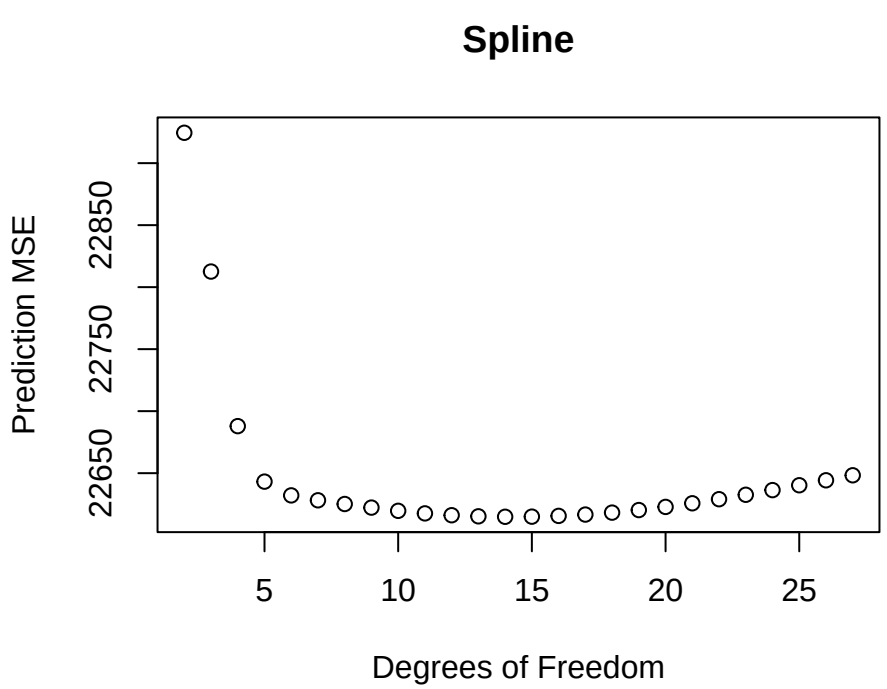
```
# Initialize the set of df's for smoothing spines
dfss <- 2:27

# Set up the storage for cross-validation
k <- 5
thefolds <- sample(rep(1:k, length=nrow(housing)), replace=FALSE)
```

Now we perform cross-validation:

```
predss <- matrix(NA, k, length(dfss))

for (fold in 1:k) {
  train <- housing[thefolds != fold, ]
  test <- housing[thefolds == fold, ]
  for (j in seq_along(dfss)) {
    ssfit <- smooth.spline(train$Median_house_value ~
                           train$Mean_household_income,
                           df=dfss[j])
    predss[fold,j] <- mean((test$Median_house_value -
                            predict(ssfit, x=test$Mean_household_income)$y)^2)
  }
}

# Compute average MSE of prediction and estimated standard errors
predssa <- apply(predss, 2, mean)

sdss <- apply(predss, 2, sd) / sqrt(k)
```

Next, we create a plot of estimated MSE against smoothing parameter:

```
plot(dfss, predssa, xlab="Degrees of Freedom",
     ylab="Prediction MSE", main="Spline")
```



And the smoothing parameter with the best MSE is

```
dfssm <- dfss[which.min(predssa)]
dfssm
## [1] 14
```

Notice that the estimated prediction MSE's vary by a few hundred from the best to the worst `df`, but the estimated standard errors are all around 200. It would not be surprising to find that we got a different "best" `df` with a different set of folds. As long as we stay away from the really small `df` values, the estimated prediction MSE doesn't change by much.

(c) Next, perform a similar analysis for kernel regression with a normal (Gaussian) kernel. Use the `npreg` function in `library(np)` to fit the kernel regressions.

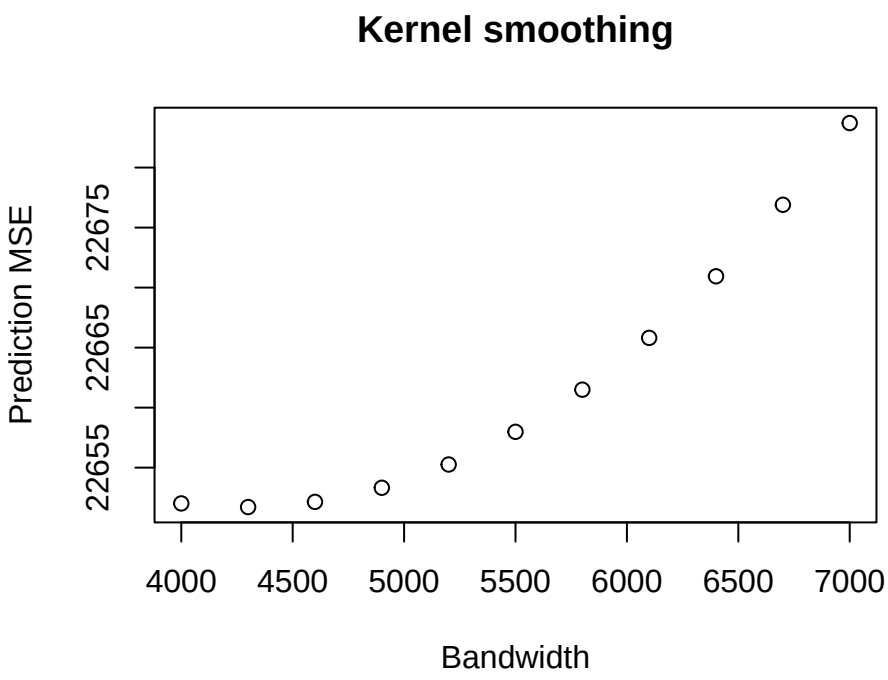
This time, there is no `df` option for the smoothing parameter. Instead, use the `bws` parameter to set the bandwidth to each of the values from 4000 to 7000 in steps of 300. (That should be 11 different values, for a sanity check.) **Make sure that you use the same five folds for five-fold cross-validation as you used in part (b). The comparisons will not be fair if you use different folds.** In the instructions from part (b), replace `df` by `bws` and repeat all of the analyses for the kernel regression models.

Solution: Once again, we initialize the smoothing parameter and run cross-validation.

```
# Initialize the set of bandwidths for kernel regression
bws <- seq(4000, 7000, by=300)

# Perform cross-validation for kernel models
predkr <- matrix(NA, k, length(bws))

for (fold in 1:k) {
  train <- housing[thefolds != fold, ]
  test <- housing[thefolds == fold, ]
  for (j in seq_along(bws)) {
    krfit <- npreg(Median_house_value ~ Mean_household_income,
                   data=train, bws=bws[j])
    predkr[fold,j] <- mean((test$Median_house_value -
                            predict(krfit, newdata=test))^2)
  }
}

# Compute average MSE of prediction and estimated standard errors
predkra <- apply(predkr, 2, mean)

sdkr <- apply(predkr, 2, sd) / sqrt(k)

# Smoothing parameter with the best MSE
bwkrm <- bws[which.min(predkra)]
```

We can plot these estimated MSEs against the smoothing bandwidth:

```
plot(bws, predkra, xlab="Bandwidth", ylab="Prediction MSE",
     main="Kernel smoothing")
```



The smoothing parameter with the best estimated MSE is 4300. Once again, the estimated standard errors are all around 209 or more, while the differences between the estimated MSE's are all less than 100. We would not be surprised to get a different "best" bandwidth with a different choice of folds.

(d) For linear smoothers of the form $\hat{Y} = LY$, the effective degrees of freedom is simply trace(L), the sum of the diagonal elements in the smoothing matrix. In the matrix multiplication, the diagonal elements give the weight of Y_i in predicting $\hat{Y}_i$.

Kernel smoothers are linear smoothers, so for a kernel smoother with bandwidth h, the effective degrees of freedom is

$$\begin{aligned}
\mathrm{df}(h) &= \mathrm{trace}(L) \\
&= \sum_{i=1}^{n} L_{ii} \\
&= \sum_{i=1}^{n} \frac{K([x_i - x_i]/h)}{\sum_{j=1}^{n} K([x_i - x_j]/h)} \\
&= K(0) \sum_{i=1}^{n} \frac{1}{\sum_{j=1}^{n} K([x_i - x_j]/h)}.
\end{aligned}$$

In our case, K is a Gaussian density function, since that's the kernel we used in (c). In R, you can calculate the Gaussian density with `dnorm`.

Write an R function to calculate df(h). Since we fit a univariate model using `Mean_household_income` as x, the function will need to take mean household income and the bandwidth h as arguments. It will loop over each data point x_i; for each entry in the loop, you'll have to calculate $\sum_{j=1}^{n} K([x_i - x_j]/h)$.

For each bandwidth (`bws`) in part (c), compute the corresponding effective degrees-of-freedom. Draw a single plot that contains the pairs

(effective degrees-of-freedom, estimated MSE of prediction)

for both the spline fits and the kernel fits, labeled well enough to be able to tell which are which. Comparing the differences between the estimated MSE's for splines and kernels to the estimated standard errors, say why this comparison does or does not inspire confidence that we can tell which method provides better out-of-sample prediction.

Solution: First, let's write a function that computes the diagonal elements of the L "hat" matrix for a kernel regression and adds them up, using the formula given above.

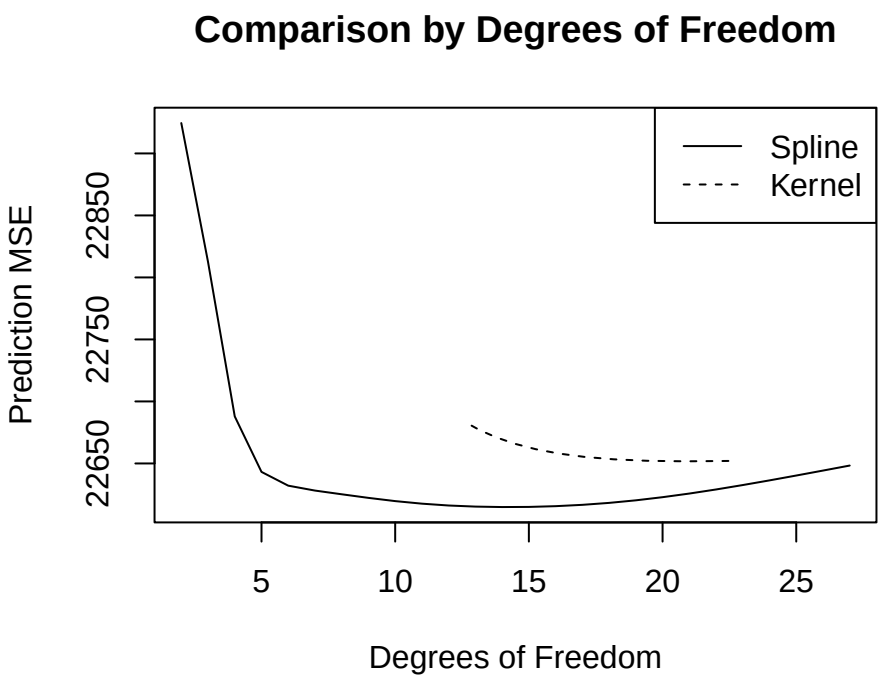
```
kdf <- function(x, h) {
  n <- length(x)
  s <- 0
  for (j in 1:n) {
    s <- s + 1 / sum(dnorm((x - x[j])/h))
  }
  return(s * dnorm(0))
}
```

Let's use this function for each bandwidth we tried.

```
dfkr <- numeric(length(bws))
for(j in seq_along(bws)) {
  dfkr[j] <- kdf(housing$Mean_household_income, bws[j])
}

# Find effective degrees-of-freedom with best MSE
dfkrm <- dfkr[which.min(predkra)]
```

Next, we plot the estimated MSEs for both spline and kernel smoothing:

```
thetop <- max(c(predkra,predssa))
thebottom <- min(c(predkra,predssa))
dfmin <- min(c(dfkr,dfss))
dfmax<- max(c(dfkr,dfss))

plot(c(dfmin,dfmax), c(thetop, thebottom), pch="",
     xlab="Degrees of Freedom", ylab="Prediction MSE",
     main="Comparison by Degrees of Freedom")
lines(dfss, predssa, lty=1)
lines(dfkr, predkra, lty=2)
legend("topright", lty=1:2, legend=c("Spline", "Kernel"))
```



The values of effective degrees-of-freedom are decreasing as functions of the bandwidth, as we would have expected. Larger bandwidths make it harder to track the data. The effective degrees-of-freedom that correspond to the bandwidths that we used are between 12.59 and 22.46, much closer together than the ones that we used for fitting splines.

The effective degrees-of-freedom that corresponds to the bandwidth with smallest estimated prediction MSE is 20.8. Over the small range of effective degrees-of-freedom that we used, the spline fits are uniformly better than the kernel fits in terms of cross-validation error. The differences between the two are still small compared to the estimated standard errors, but they are about four times the size of the differences between the group of best spline fits.

(e) Finally redraw the plot from part (a) and add

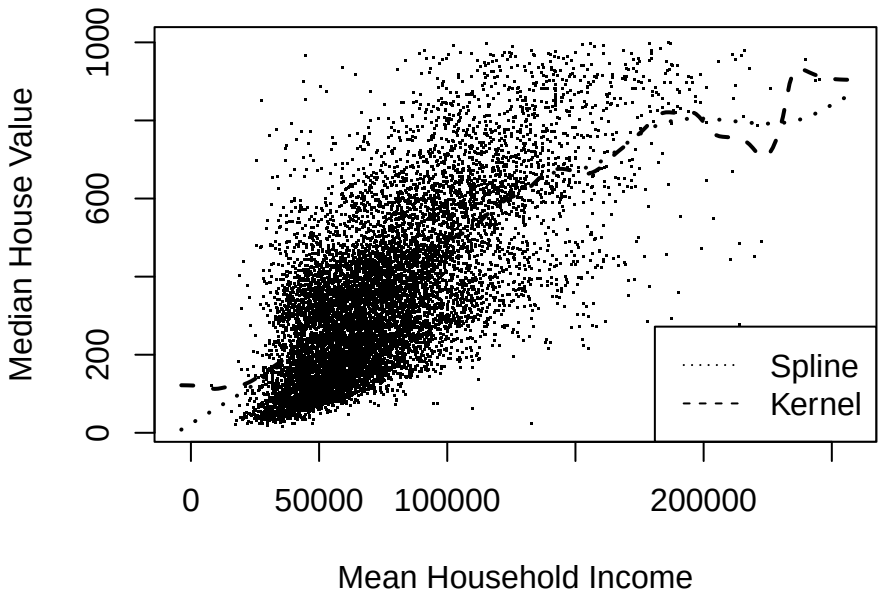
(i) a line for the spline fit corresponding to the best smoothing parameter value, and

(ii) a line for the kernel regression fit with the best smothing parameter value.

For the two lines, use a common set of 201 predictor values that is equally-spaced over the observed range of `Mean_household_income` and extending 5% past each end. Comment on how well or badly each of the two fitted curves seems to fit the data. Also offer a reason for why the two fitted curves behave so differently at and beyond the extremes of the predictor.

Solution: The redrawn plot appears below with the curves added. There appear to be two "clouds" of data points in the plot, one right above the two fitted curves and one right below. The curves are nearly identical throughout much of the range of the data, diverging from each other near the extremes as one would expect. Kernel regressions with a Gaussian kernel eventually flatten out at the response value corresponding to the most extreme observation at each end. Smoothing splines become linear with a slope that approximates the general direction in which the points are moving at each end.

```
# Set up the common set of predictors for the plots
therange <- range(housing$Mean_household_income)
thelength <- therange[2] - therange[1]
x0 <- seq(therange[1] - .05 * thelength, therange[2] + 0.05 * thelength,
          length.out = 201)

# Plot data and best fitting curves
plot(x0[c(1, 201)], range(housing$Median_house_value),
     pch = "",
     xlab = "Mean Household Income", ylab = "Median House Value")
points(housing$Mean_household_income,
       housing$Median_house_value,
       pch = ".")
krmin <- npreg(Median_house_value ~ Mean_household_income,
               data = housing, bws = bwkrm)
ssmin <- smooth.spline(housing$Median_house_value ~ housing$Mean_household_income,
                       df = dfssm)
lines(x0, predict(ssmin, x = x0)$y, lty = 3, lwd = 2)
lines(x0, predict(krmin, newdata = data.frame(Mean_household_income = x0)),
      lty = 2, lwd = 2)
legend("bottomright", lty = c(3, 2), legend = c("Spline", "Kernel"))
```



2. **Cat Data: The Sequel.** To load the data, run `library(MASS)`. We looked at the cat data during Lecture 4B and in the `BootstrapRegressionExample.Rmd` demo, so refer back to that for context on the data and variables.

   (a) Fit a linear regression model for `Hwt`, in which `Bwt` interacts with `Sex` and the intercept is forced to be zero. Show the model summary, and explain why forcing the intercept to be zero is a

reasonable thing to do.

Hint: There are some subtleties regarding how to force the intercept to be zero. Make sure you are not accidentally introducing intercept terms by using the wrong `R` syntax.

Solution:

```
library(MASS)
##
## Attaching package:  'MASS'
## The following object is masked _by_ '.GlobalEnv':
##
##     housing
fit <- lm(Hwt ~ Bwt:Sex - 1, data = cats)
summary(fit)
##
## Call:
## lm(formula = Hwt ~ Bwt:Sex - 1, data = cats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4841 -0.9929 -0.1036  0.9879  5.2330
##
## Coefficients:
##          Estimate Std. Error t value Pr(>|t|)
## Bwt:SexF  3.88345    0.08925   43.51   <2e-16 ***
## Bwt:SexM  3.91461    0.05024   77.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.453 on 142 degrees of freedom
## Multiple R-squared:  0.9825,Adjusted R-squared:  0.9822
## F-statistic:  3982 on 2 and 142 DF,  p-value: < 2.2e-16
```

Forcing the intercept to be zero makes sense because we would expect that if body weight is equal to 0 so is heart weight.

We can interpret the coefficients of this model as giving the slopes separately for male and female cats; you can see the slopes are quite similar, though not exactly the same.

(b) Under the assumption that the residuals are normally distributed, conduct a statistical test to find out whether there is evidence that the slope coefficient for `Bwt` differs between female and male cats. Make sure you state the hypotheses (null and alternative), the null distribution, the value of the test statistic and the p value.

Hint: Find a way to write the hypothesis test as the comparison of two different linear models, and recall tests you might have used in 36-401 to compare linear models.

Solution: There are many parametrizations of the null and alternative models for this test. For example, we could have

$$H_0 : \mathbb{E}(Y|\texttt{Bwt}, \texttt{Sex}) = \alpha_1 \texttt{Bwt}$$
$$H_1 : \mathbb{E}(Y|\texttt{Bwt}, \texttt{Sex}) = \alpha_1 \texttt{Bwt} + \alpha_2 \texttt{Bwt} \cdot \mathbb{I}\{\texttt{Sex} = \texttt{Female}\}$$

so we would be interested in testing whether $\alpha_2 = 0$. Alternatively, we could specify models

$$H_0 : \mathbb{E}(Y|\texttt{Bwt}, \texttt{Sex}) = \alpha_1 \texttt{Bwt}$$
$$H_1 : \mathbb{E}(Y|\texttt{Bwt}, \texttt{Sex}) = \alpha_1 \texttt{Bwt} \cdot \mathbb{I}\{\texttt{Sex} = \texttt{Male}\} + \alpha_2 \texttt{Bwt} \cdot \mathbb{I}\{\texttt{Sex} = \texttt{Female}\}$$

Notice that $\mathbb{I}\{\texttt{Sex} = \texttt{Male}\} = 1 - \mathbb{I}\{\texttt{Sex} = \texttt{Female}\}$:

$$\begin{aligned} H_0 : \mathbb{E}(Y|\texttt{Bwt}, \texttt{Sex}) &= \alpha \texttt{Bwt} \\ H_1 : \mathbb{E}(Y|\texttt{Bwt}, \texttt{Sex}) &= \alpha_1 \texttt{Bwt} \cdot \mathbb{I}\{\texttt{Sex} = \texttt{Male}\} + \alpha_2 \texttt{Bwt} \cdot \mathbb{I}\{\texttt{Sex} = \texttt{Female}\} \\ &= \alpha_1 \texttt{Bwt} + (\alpha_2 - \alpha_1)\texttt{Bwt} \cdot \mathbb{I}\{\texttt{Sex} = \texttt{Female}\} \end{aligned}$$

So, the two models are exactly the same, but only one parametrization makes the null model nested in the alternative model. Testing using the second set of models can be done via the `linearHypothesis` function in R. Alternatively, you could appeal to asymptotic normality of $\hat{\alpha}$ and compare the test statistic $\frac{\hat{\alpha_1} - \hat{\alpha_2}}{\sqrt{\hat{\text{Var}}(\hat{\alpha_1} - \hat{\alpha_2})}}$ to a standard normal distribution.

We'll use the first parametrization and use `anova` to test the hypotheses:

```
fit_null <- lm(Hwt ~ 0 + Bwt, data = cats)
fit_alt <- lm(Hwt ~ 0 + Bwt + Bwt:Sex, data = cats)
anova(fit_null, fit_alt)
## Analysis of Variance Table
##
## Model 1: Hwt ~ 0 + Bwt
## Model 2: Hwt ~ 0 + Bwt + Bwt:Sex
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    143 300.09
## 2    142 299.90  1   0.19544 0.0925 0.7614
```

The value of the t-statistic is 0.304, which, compared to a the t-distribution with $n - 1 = 142$ degrees of freedom, yields a p-value ≈ 0.76. Therefore, we fail to reject the null hypothesis that the slope for `Bwt` differs between males and females cats.

(c) Now let's test the same hypothesis *without* making assumptions about normality of the residuals. As you know, statistical hypothesis tests—like the one you did in part (b)—work by comparing a statistic calculated from the data to the distribution of that statistics expected when the null hypothesis is true. We choose a statistic that measures differences from what we expect under the null hypothesis, so in this case we will use

$$T = (\hat{\beta}_{\texttt{Bwt:Male}} - \hat{\beta}_{\texttt{Bwt:Female}})^2.$$

To calculate the statistic on the observed data, use the model you fit in part (a). Report the value of the test statistic and save it in a variable for use later.

Solution: We get

```
tobs <- (coefficients(fit)[1] - coefficients(fit)[2])**2
tobs
##     Bwt:SexF
## 0.0009707399
```

(d) Now we must estimate the distribution of statistics expected when the null hypothesis is true (the "null distribution"). Because we are not assuming the residuals are normal, it is not obvious what

the sampling distributions of $\hat{\beta}_{\texttt{Bwt:Male}}$ and $\hat{\beta}_{\texttt{Bwt:Female}}$ are, or what the sampling distribution of T is.

Instead, we will use the bootstrap by resampling residuals to simulate the distribution of T when $\hat{\beta}_{\texttt{Bwt:Male}} = \hat{\beta}_{\texttt{Bwt:Female}}$. When we bootstrap by resampling residuals, we draw new observations (x_i, Y_i^*) using

$$Y_i^* = \hat{r}(x_i) + \epsilon_i^*,$$

where ϵ_i^* is drawn at random and with replacement from the observed residuals. This is like drawing new observations from a population where $\hat{r}(x)$ is the true population regression function. If we want to obtain a distribution when the null hypothesis is true, we can simply insert the correct $\hat{r}(x)$ so that

$$\hat{r}_{\text{Male}}(x) = \hat{r}_{\text{Female}}(x) = \hat{\beta}_1 x,$$

where $\hat{\beta}_1$ is the slope you calculate from a regression that ignores `Sex`.[1] When we bootstrap in this way, we obtain samples where the null hypothesis is true.

Write an R function that produces bootstrap samples from the data, but using this model. In case the residual distribution is different for male and female cats, it should draw the residuals for male cats from the observed residuals for male cats, and the residuals for female cats from the observed residuals for female cats.

Next, for each bootstrap sample b, let T_b^* denote the value of T computed from sample b. Run $B = 1000$ bootstrap samples and plot a histogram of T_b^*. Mark the T statistic you calculated in (c) as a vertical line on the histogram.

Solution: First, the function for bootstrapping:

[1] It is possible to prove that the bootstrap by resampling residuals will lead to the same results no matter what $\hat{\beta}_1$ you use for computing $\hat{r}_{\text{Male}}(x) = \hat{r}_{\text{Female}}(x)$. All that matters is that you use the same $\hat{\beta}$s for both.

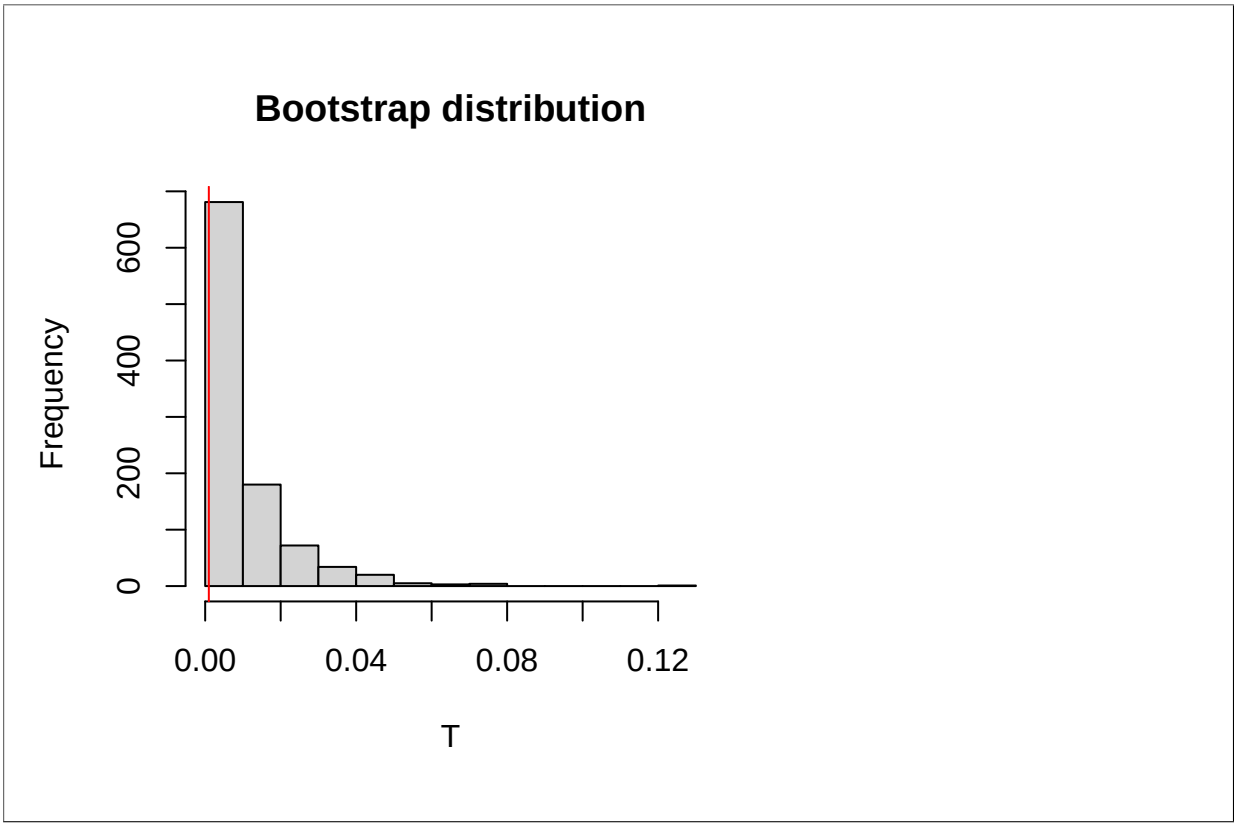
```r
# Regression ignoring sex
catlm1 <- lm(Hwt ~ 0 + Bwt, data = cats)

# Separate regressions
catlmF <- lm(Hwt ~ 0 + Bwt, data = cats[cats$Sex == "F",])
catlmM <- lm(Hwt ~ 0 + Bwt, data = cats[cats$Sex == "M",])

nf <- sum(cats$Sex == "F")
nm <- sum(cats$Sex == "M")

bootstrap_cats <- function() {
  # Simulate the noise terms for females and males by bootstrapping
  # residuals. For females, sample specifically from the residuals
  # observed for female cats; similarly for males, because we noted in (a)
  # that the residual distributions don't seem to be the same.
  fnoise <- residuals(catlmF)[sample(sum(nf), replace=T)]
  mnoise <- residuals(catlmM)[sample(sum(nm), replace=T)]

  # Create the bootstrap data sets by adding the residuals to r(x),
  # where r(x) is the same for male and female cats.
  newfemales <- data.frame(
    Bwt = cats$Bwt[cats$Sex=="F"],
    Hwt = catlm1$coef[1] * cats$Bwt[cats$Sex=="F"] + fnoise,
    Sex = "F")

  newmales <- data.frame(
    Bwt = cats$Bwt[cats$Sex=="M"],
    Hwt = catlm1$coef[1] * cats$Bwt[cats$Sex=="M"] + mnoise,
    Sex = "M")

  return(rbind(newfemales, newmales))
}
```

Next we run it 1000 times:

```r
B <- 1000
Tstar <- numeric(B)

for (b in 1:B) {
  new_cats <- bootstrap_cats()

  new_fit <- lm(Hwt ~ Bwt:Sex - 1, data = new_cats)

  Tstar[b] <- (coefficients(new_fit)[1] - coefficients(new_fit)[2])**2
}
```

And finally, we make our histogram:

```r
hist(Tstar, xlab = "T", main = "Bootstrap distribution")
abline(v = tobs, col = "red")
```



(e) Recall that the p-value is defined by

$$\Pr(T \geq t_{\text{obs}} \text{ when } H_0 \text{ is true}), \tag{1}$$

where t_{obs} is the observed value of T. Bootstrap the p value, that is, compute $\Pr(T^* \geq t_{\text{obs}})$ as an estimate of (1).

Based on your result, is there a significant difference between the regression lines for male and female cats?

Solution: The p value is just the fraction of the time that the observed value is greater than the bootstrapped null values:

```
pvalue <- mean(Tstar >= tobs)
pvalue
## [1] 0.754
```

The p-value is the probability in the upper tail of the distribution of T^* (supposedly the bootstrap distribution of T_b^*) at or above the observed statistic $t_{\text{obs}} = 9.707 \times 10^{-4}$. From output above, we see that the p-value is 0.754. This isn't too different from what we got from `anova`. If we set $\alpha = 0.05$, we would not reject the null hypothesis.