36-402 Homework 7

James “Morgan” Hawkins

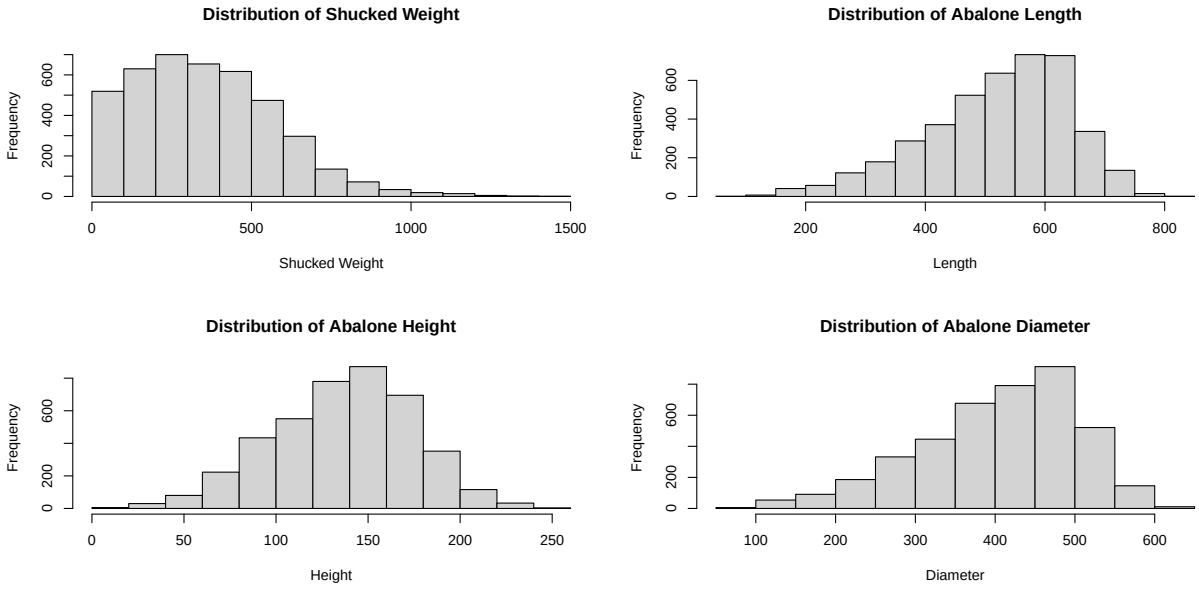
Friday April 1, 2023

Problem 1

Problem 1 (a)

```
abalone <- read.csv("/Users/morganhawkins/downloads/abalonemt.csv")
# dim(abalone)
# head(abalone)
```

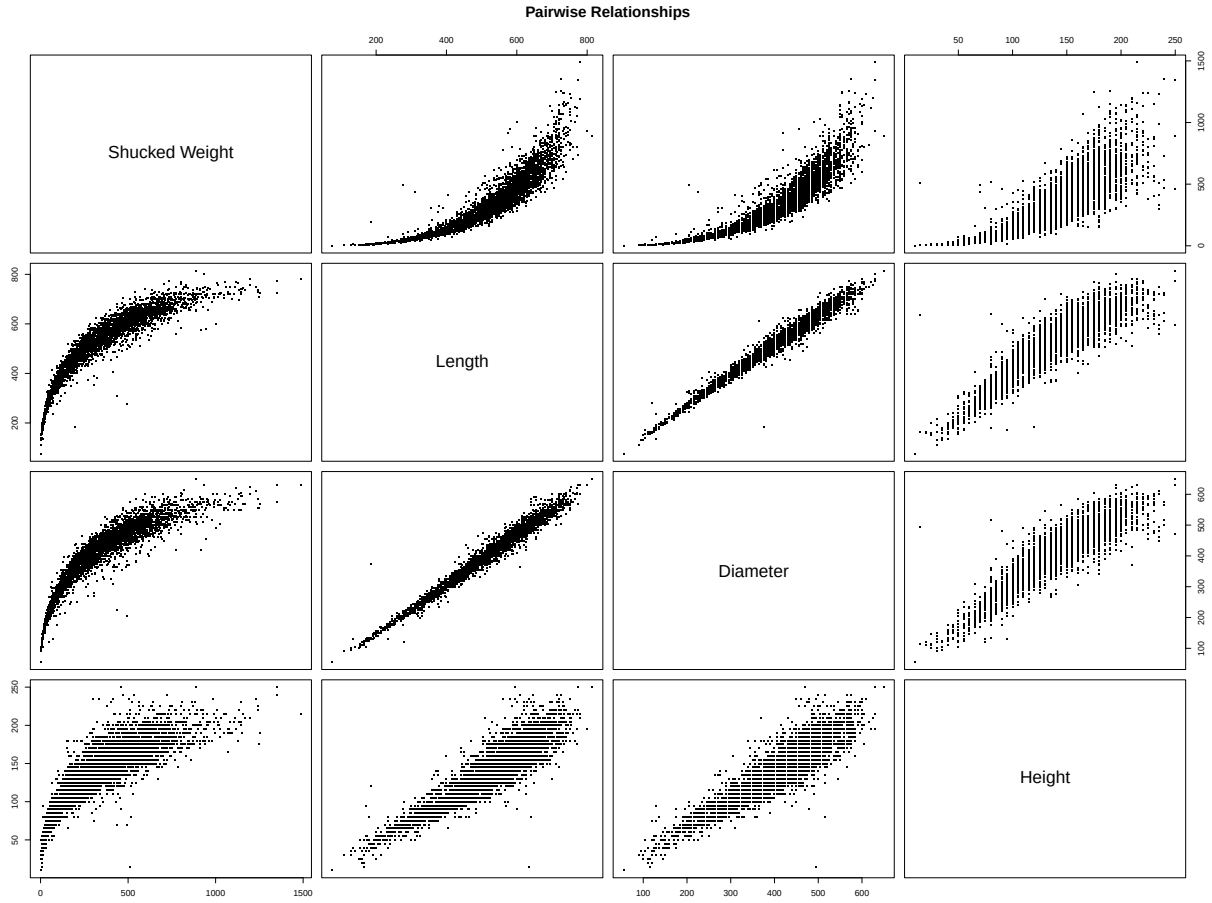
```
par(mfrow = c(2,2))

hist(abalone$Shucked.weight, main = "Distribution of Shucked Weight",
     xlab = "Shucked Weight")
hist(abalone$Length, main = "Distribution of Abalone Length",
     xlab = "Length")
hist(abalone$Height, main = "Distribution of Abalone Height",
     xlab = "Height")
hist(abalone$Diameter, main = "Distribution of Abalone Diameter",
     xlab = "Diameter")
```



```
#sapply(abalone[,-1],mean)
```

Looking at the plots above, we see that our response and all three predictors appear unimodal. Height appears to be symmetrically distributed, diameter and length appear to be left skewed, and shucked weight appears to be right skewed. Our variables of interest, shucked weight, length, diameter, and height have sample means of 259.3, 524.0, 407.9, and 139.3 respectively.

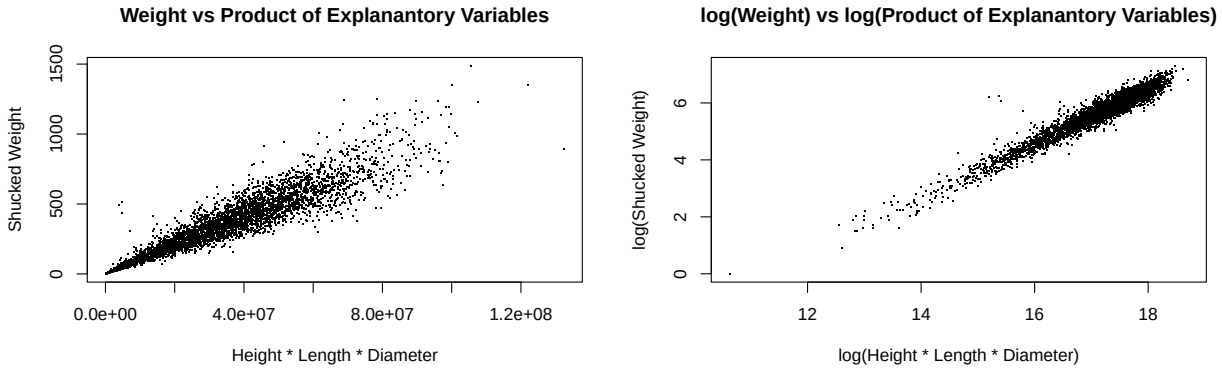
```
abalone$Shucked.weight = abalone$Shucked.weight
pairs(abalone[,c("Shucked.weight", "Length", "Diameter", "Height")],
      main = "Pairwise Relationships", pch = '.',
      labels = c("Shucked Weight", "Length", "Diameter", "Height"))
```



In the plot above we see that our explanatory variables all appear to have strong positive linear relationships. Because of this, combining explanatory variables may be be beneficial. We know weight is proportional to volume when an object has uniform density, so we expect shucked weight to have a lienar relationship with the product of all three explanantory variables. We also see that the response appears to have a nonlinear positive relationship with all three predictors. So, transformations may be beneficial to this model.

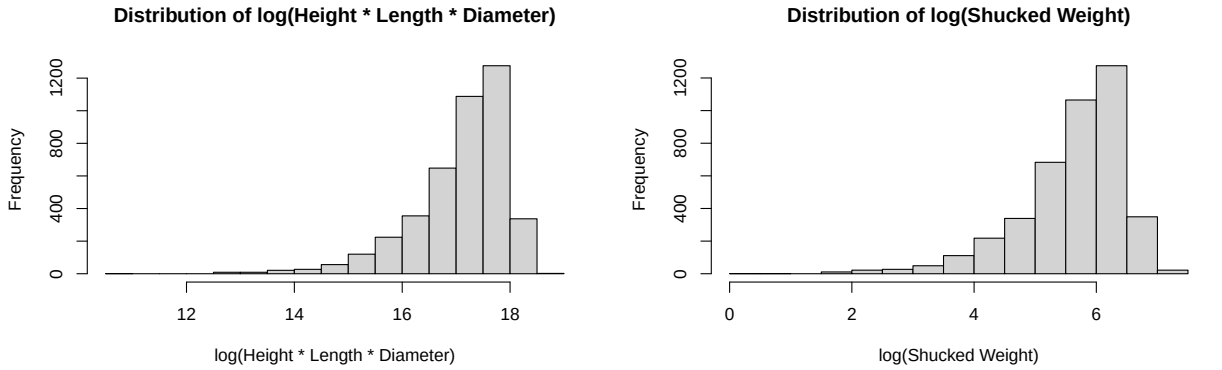
```
par(mfrow = c(1,2))
plot(abalone$Length*abalone$Height*abalone$Diameter, abalone$Shucked.weight,
     main = "Weight vs Product of Explanantory Variables",
     xlab = "Height * Length * Diameter",
     ylab = "Shucked Weight",
```

```
     pch = '.')
plot(log(abalone$Length*abalone$Height*abalone$Diameter), log(abalone$Shucked.weight),
     main = "log(Weight) vs log(Product of Explanantory Variables)",
     xlab = "log(Height * Length * Diameter)",
     ylab = "log(Shucked Weight)",
     pch = '.')
```



Above, we see that when we plot the product of the explanatory variables against shucked weight, our relationship still does not appear to be linear. However, when we log transform both sides of our regression function, we see the the relationship appears to be linear. We also see that our residuals appear more similarly distributed which is desirable.

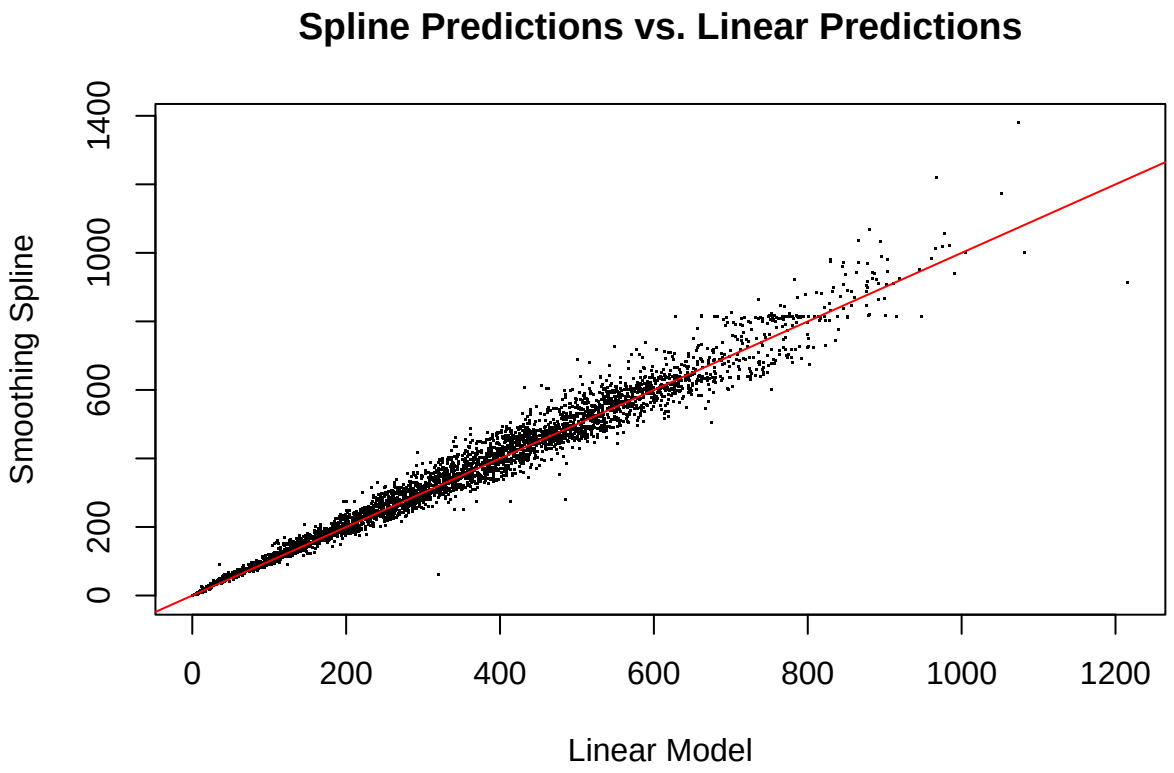
```
par(mfrow = c(1,2))

hist(log(abalone$Length*abalone$Height*abalone$Diameter),
     main = "Distribution of log(Height * Length * Diameter)",
     xlab = "log(Height * Length * Diameter)")
hist(log(abalone$Shucked.weight),
     main = "Distribution of log(Shucked Weight)",
     xlab = "log(Shucked Weight)")
```



Lastly, we see that our transformed explanatory variable and response both appear to have left-skewed unimodal distributions. Both variables do not appear normally distributed.

Problem 1 (b)

```
linear.model = lm(log(Shucked.weight) ~ log(Height) + log(Diameter) + log(Length),
                  data = abalone)
#linear.model = lm(log(Shucked.weight) ~ log(Height*Diameter*Length), data = abalone)
#linear.model %>% summary

spline.model = smooth.spline(abalone$Shucked.weight ~
                               abalone$Length*abalone$Height*abalone$Diameter)
#mean(residuals(spline.model)^2)^.5

prediction.df = data.frame(x = abalone$Length*abalone$Height*abalone$Diameter,
                           y = abalone$Shucked.weight,
                           linear = exp(fitted(linear.model)),
                           spline = fitted(spline.model))

plot(prediction.df$linear, prediction.df$spline,
     main = "Spline Predictions vs. Linear Predictions",
     xlab = "Linear Model",
     ylab = "Smoothing Spline",
     pch = '.')
abline(0,1,col = 'red')
```



Problem 1 (c)

```
mean((exp(fitted(linear.model)) - abalone$Shucked.weight)^2)
```

```
## [1] 5393.721
```

```
mean((fitted(spline.model) - abalone$Shucked.weight)^2)
```

```
## [1] 6408.51
```

Our linear model has a training error of 5393.72 while our smoothing splines has a training error of 6408.51, so we see that our linear model fits the training data better.

Problem 1 (d)

```
#5-fold CV

set.seed(99)
fold.size = round(dim(abalone)[1]/5)
folds = sample(dim(abalone)[1], replace = FALSE)

linear.cv.mse = c()
spline.cv.mse = c()

#tic()
for(f in 0:4){
  test.start = f*fold.size + 1
  test.end = min((f+1)*fold.size, dim(abalone)[1])

  test.data = abalone[folds[test.start:test.end],c(2,3,4)]
  train.data = abalone[-folds[test.start:test.end],c(2,3,4,6)]
  test.response = abalone[folds[test.start:test.end],6]
  test.explan = test.data$Length*test.data$Height*test.data$Diameter

  cv.linear = lm(log(Shucked.weight) ~ log(Height) + log(Diameter) + log(Length),
                 data = train.data)
  cv.spline = smooth.spline(train.data$Shucked.weight ~
                              (train.data$Length*train.data$Height*train.data$Diameter))

  cv.linear.resids = exp(predict(cv.linear, newdata = test.data)) - test.response
  cv.spline.resids = predict(cv.spline, x = test.explan)$y - test.response

  linear.cv.mse = c(linear.cv.mse, mean(cv.linear.resids^2))
  spline.cv.mse = c(spline.cv.mse, mean(cv.spline.resids^2))



}
#toc()
```

```
data.frame(linear.model = c("mean" = mean(linear.cv.mse),
                            "st err" = sd(linear.cv.mse)/sqrt(5)),
           smoothing.spline = c(mean(spline.cv.mse),
                                sd(spline.cv.mse)/sqrt(5))) %>% (knitr::kable)
```

	linear.model	smoothing.spline
mean	5415.1992	6884.6915
st err	206.4145	225.2802

Based on these cross validation results, the linear model appears to perform better as it has a lower test error as well as a lower standard error on this estimate.

Problem 1 (e)

```
#summary(linear.model)
0.83326 + qnorm(.025)*0.07140
```

```
## [1] 0.6933186
```

```
0.83326 + qnorm(.975)*0.07140
```

```
## [1] 0.9732014
```

```
confint(linear.model)[c(3,7)]
```

```
## [1] 0.6932753 0.9732489
```

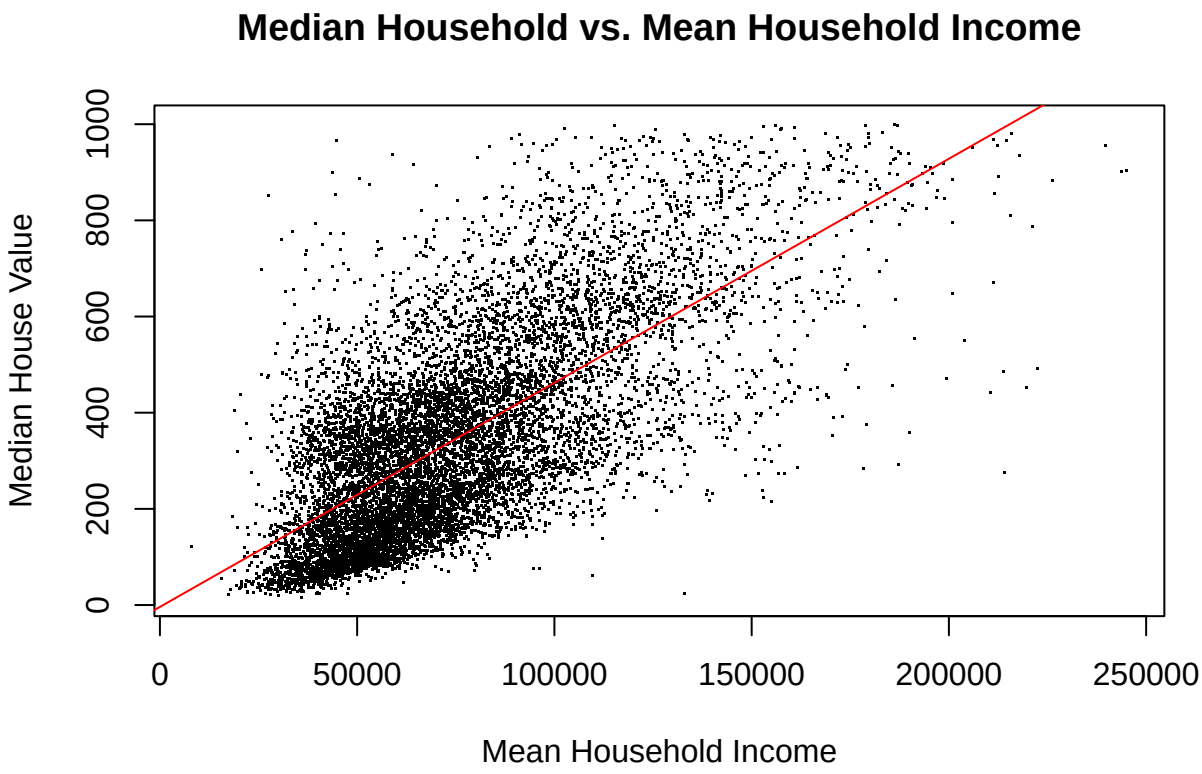
Under the assumption that $\hat{\beta}$ from the linear model is multivariate normal, our 95% confidence interval for $\beta_{log-Diameter}$ is [.693, .973]. No, it is not reasonable to assume this because our predictors are significantly correlated.

Problem 2

```
house = read.csv("/Users/morganhawkins/Downloads/house.csv")
```

Problem 2 (a)

```
plot(house$Mean_household_income, house$Median_house_value, pch = '.',
     main = "Median Household vs. Mean Household Income",
     xlab = "Mean Household Income",
     ylab = "Median House Value")
lm(house$Median_house_value ~ house$Mean_household_income ) %>%
  abline(col = 'red')
```

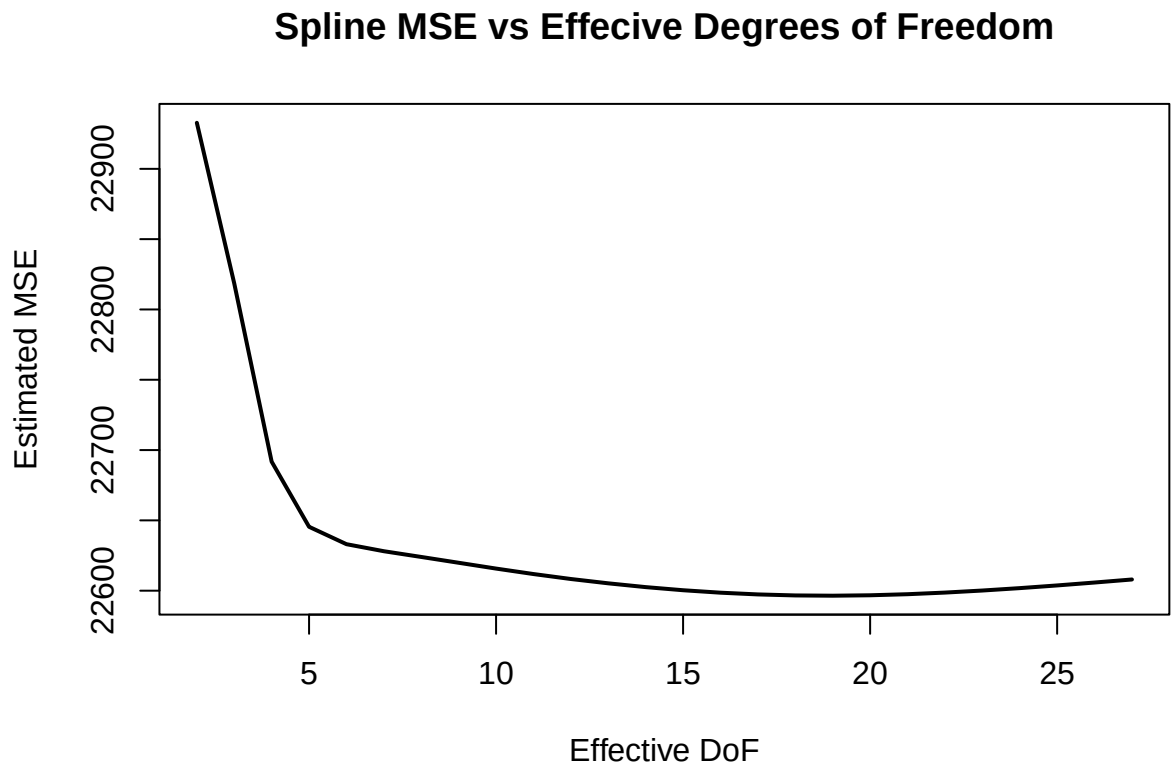


Problem 2 (b)

```
spline.models = c()

for(deg.free in 2:27){
  new.spline = smooth.spline(house$Median_house_value ~ house$Mean_household_income,
                             df = deg.free)
  spline.models = append(spline.models, new.spline)
}
```

```
set.seed(999)

fold.size = round(dim(house)[1]/5)
folds = sample(dim(house)[1], replace = FALSE)

spline.df = data.frame(met = c("mean","sd"))


for(deg.free in 2:27){

  cv.mse = c()
  for(f in 0:4){
    test.start = f*fold.size + 1
    test.end = min((f+1)*fold.size, dim(house)[1])
```

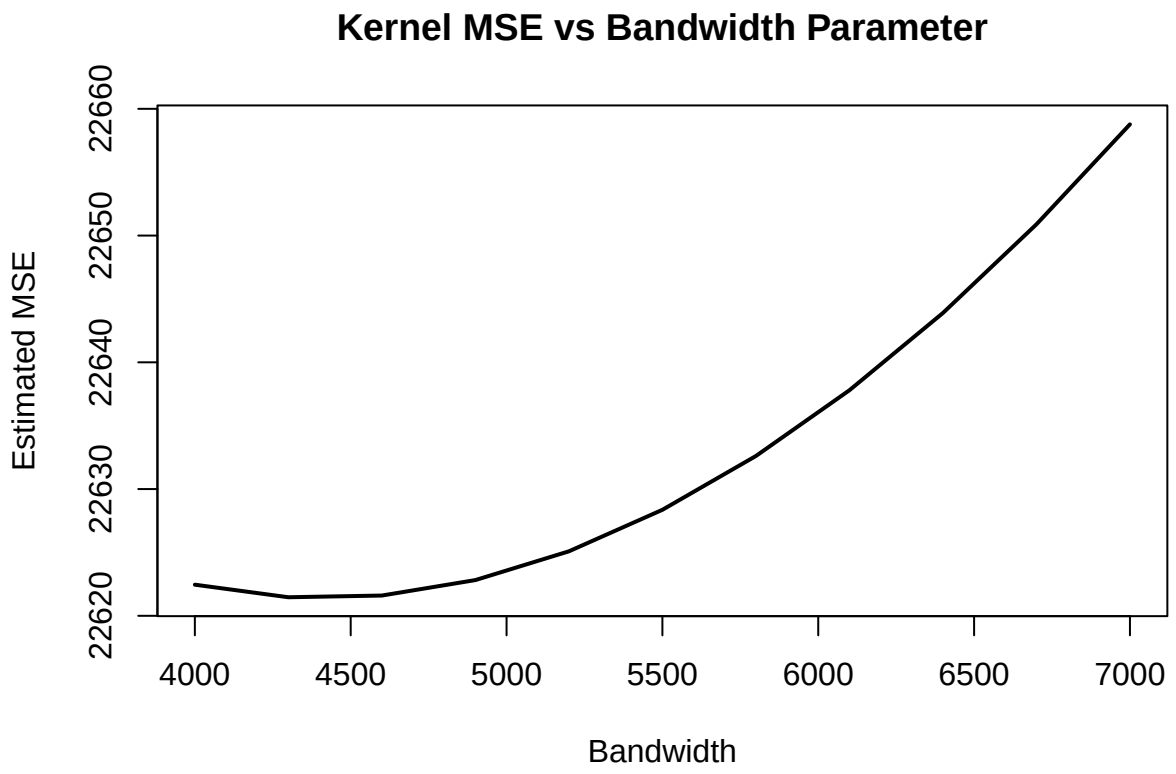
```
    train.x = house[-folds[test.start:test.end],"Mean_household_income"]
    train.y = house[-folds[test.start:test.end],"Median_house_value"]
    test.x = house[folds[test.start:test.end],"Mean_household_income"]
    test.y = house[folds[test.start:test.end],"Median_house_value"]

    cv.spline = smooth.spline(train.y ~ train.x, df = deg.free)

    test.errors = predict(cv.spline, x = test.x)$y - test.y

    cv.mse = c(cv.mse, mean(test.errors^2))

  }
  spline.df[paste("df.",deg.free)] = c(mean(cv.mse),sd(cv.mse)/sqrt(5))
}

#reformat cross-val results
spline.df = data.frame(t(spline.df)[-1,])
colnames(spline.df) = c("mean", "st.err")

#plotting results
plot(2:27, spline.df$mean, type = 'l', lwd = 2,
     xlab = "Effective DoF",
     ylab = "Estimated MSE",
     main = "Spline MSE vs Effecive Degrees of Freedom")
```



19 effective degrees of freedom provides the lowest MSE estimate. Comparing the differences in sample averages relative to their standard errors, we see that there these estimates are relatively close. The difference between the worst choice for DoF and best choice for DoF is just 354.30. This is smaller than the smallest standard error estimate which is 356.12 (associated with the MSE estimate for 27 DoF). This does not inspire confidence that we have actually found the DoF that provides the best out-of-sample accuracy.

Problem 2 (c)

```
#using folds created in 2b

kernel.df = data.frame(met = c("mean","sd"))

for(band.w in seq(4000, 7000, 300)){

  cv.mse = c()
  for(f in 0:4){
    test.start = f*fold.size + 1
    test.end = min((f+1)*fold.size, dim(house)[1])


    train = house[-folds[test.start:test.end],]
    test = house[folds[test.start:test.end],]

    cv.kernel = npreg(Median_house_value ~ Mean_household_income,
                      data = train, newdata = test, bws = c(band.w))

    test.errors = cv.kernel$mean - test[,"Median_house_value"]

    cv.mse = c(cv.mse, mean(test.errors^2))

  }
  kernel.df[paste("bw.",band.w)] = c(mean(cv.mse),sd(cv.mse)/sqrt(5))
}


#reformat cross-val results
kernel.df = data.frame(t(kernel.df)[-1,])
colnames(kernel.df) = c("mean", "st.err")

#plotting results
plot(seq(4000, 7000, 300), kernel.df$mean, type = 'l', lwd = 2,
     xlab = "Bandwidth",
     ylab = "Estimated MSE",
     main = "Kernel MSE vs Bandwidth Parameter")
```



A bandwidth parameter of 4300 produces the lowest MSE estimate. Comparing the differences in sample averages relative to their standard errors, we see that there these estimates are relatively close. The difference between the worst choice for DoF and best choice for DoF is just 36.28. This much smaller than the smallest standard error estimate which is 357.5831 (associated with the MSE estimate for 27 DoF). This does not inspire confidence that we have actually found the DoF that provides the best out-of-sample accuracy.

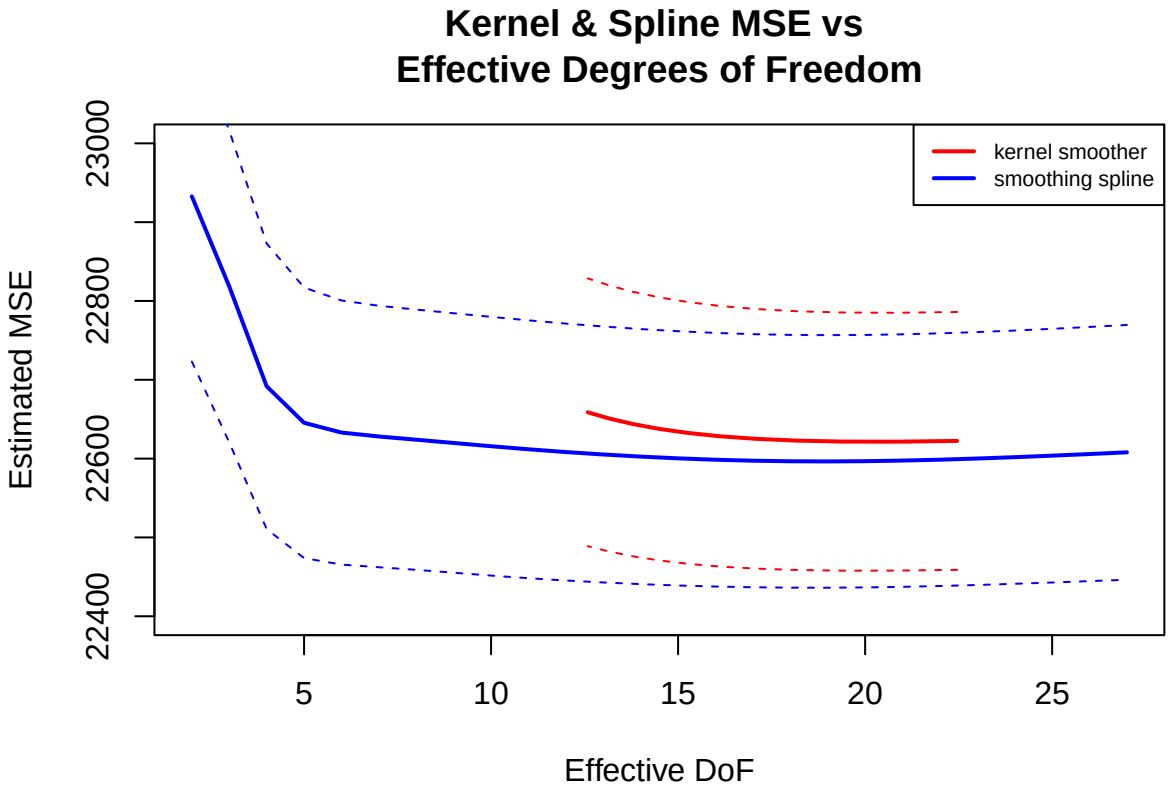
Problem 2 (d)

```
kernel.effec.df = function(h, x){

  sum = 0

  for(i in 1:length(x)){
    denom.sum = dnorm((x[i] - x)/h) %>% sum
    sum = sum + (1/denom.sum)
  }

  return(sum*dnorm(0))
}
```

```
df.from.bws = c()
for(b in seq(4000,7000, 300)){

  df.from.bws = c(df.from.bws, kernel.effec.df(b,house$Mean_household_income))
```

```
}

kernel.df$df = df.from.bws
```

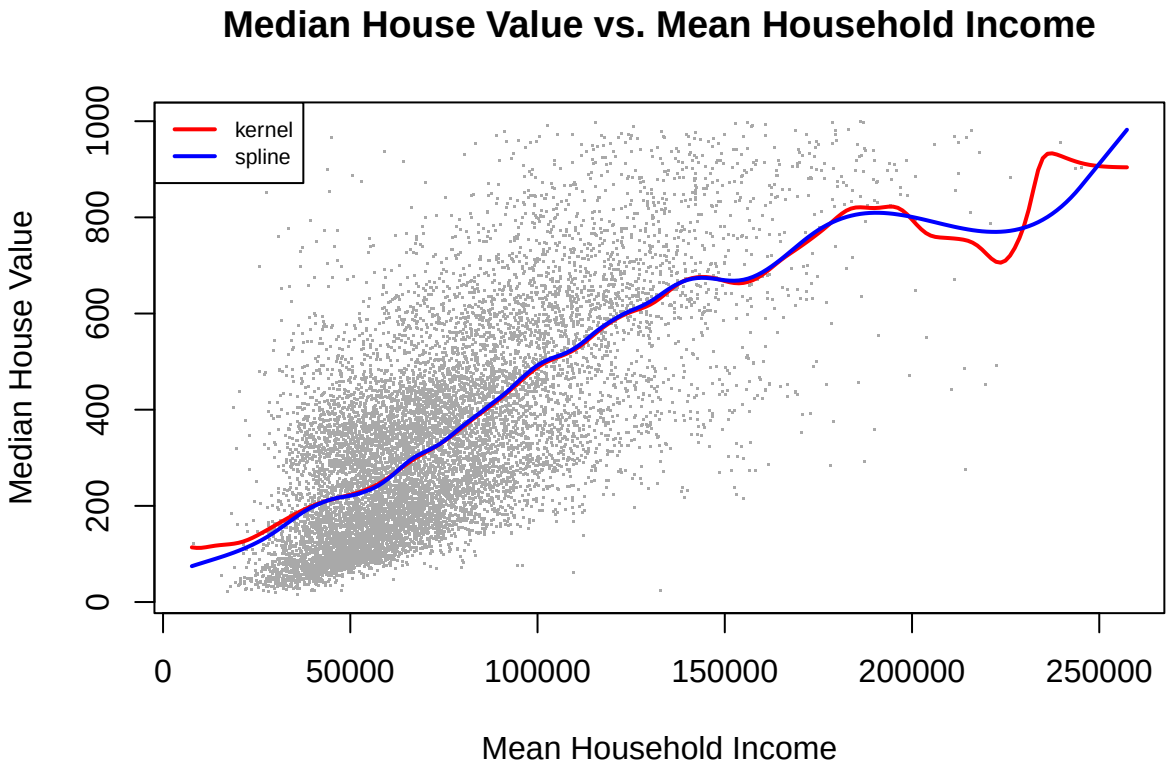
```
plot(kernel.df$df, kernel.df$mean, type = 'l', lwd = 2,
     xlab = "Effective DoF",
     ylab = "Estimated MSE",
     main = "Kernel & Spline MSE vs \nEffective Degrees of Freedom",
     ylim = c(22400, 23000),
     xlim = c(2,27),
     col = 'red')

kernel.upper = as.numeric(kernel.df$mean) + as.numeric(kernel.df$st.err)
kernel.lower = as.numeric(kernel.df$mean) - as.numeric(kernel.df$st.err)

spline.upper = as.numeric(spline.df$mean) + as.numeric(spline.df$st.err)
spline.lower = as.numeric(spline.df$mean) - as.numeric(spline.df$st.err)

lines(kernel.df$df,kernel.upper, lty = 2, col = 'red')
lines(kernel.df$df,kernel.lower, lty = 2, col = 'red')

lines(2:27,spline.upper, lty = 2, col = 'blue')
lines(2:27,spline.lower, lty = 2, col = 'blue')

lines(2:27, spline.df$mean, type = 'l', lwd = 2,
      col = 'blue')
legend(x = "topright", legend = c("kernel smoother", "smoothing spline"),
       col = c("red", "blue"), lw = 2, cex = .7)
```



On the plot above we can see that the kernel smoother and smoothing spline have similar estimated MSEs relative to the estimates' standard errors. This is shown by both MSE vs DoF curves being well within each others' range of += 1 standard error. This does not inspire confidence that one method is better than the other.

Problem 2 (e)

```
x0.lower = min(house$Mean_household_income*.95)
x0.upper = max(house$Mean_household_income*1.05)
x0 = seq(x0.lower, x0.upper, length.out = 201)

kernel.preds = predict(kernel.optim, newdata = data.frame(Mean_household_income = x0))
spline.preds = predict(spline.optim, x = x0)

plot(house$Mean_household_income, house$Median_house_value, pch = '.', col = 'darkgrey',
     main = "Median House Value vs. Mean Household Income",
     xlab = "Mean Household Income",
     ylab = "Median House Value",
     xlim = c(x0.lower, x0.upper))
lines(x0,kernel.preds, col = 'red', lwd = 2)
lines(x0,spline.preds$y, col = 'blue', lwd = 2)
#lm(house$Median_house_value ~ house$Mean_household_income) %>% abline(.,col = 'darkgreen')
legend(x = 'topleft', legend = c("kernel", "spline"), col = c('red','blue'), lw = 2, cex = .7)
```



On the plot above we can see that the spline and kernel smoother give very similar predictions for explanatory value between 5000 and 18,000. However, at the tails of the plot we see that the kernel produces more extreme values than the spline. The kernel appears to be over fitting at the tails more than the spline.