YK_Assignment10

Yinan Kang

4/26/2019

```
require(genridge)
require(lmridge)
require(rpart)
require(boot)
```

Problem 11.8

```
# Import Data
rm(list=ls())
colnames <- c("y","degree","x3","x4")
df <- read.table(url("http://users.stat.ufl.edu/~rrandles/sta4210/Rclassnotes/data/textdatasets/KutnerDa
n <- nrow(df)
attach(df)
```

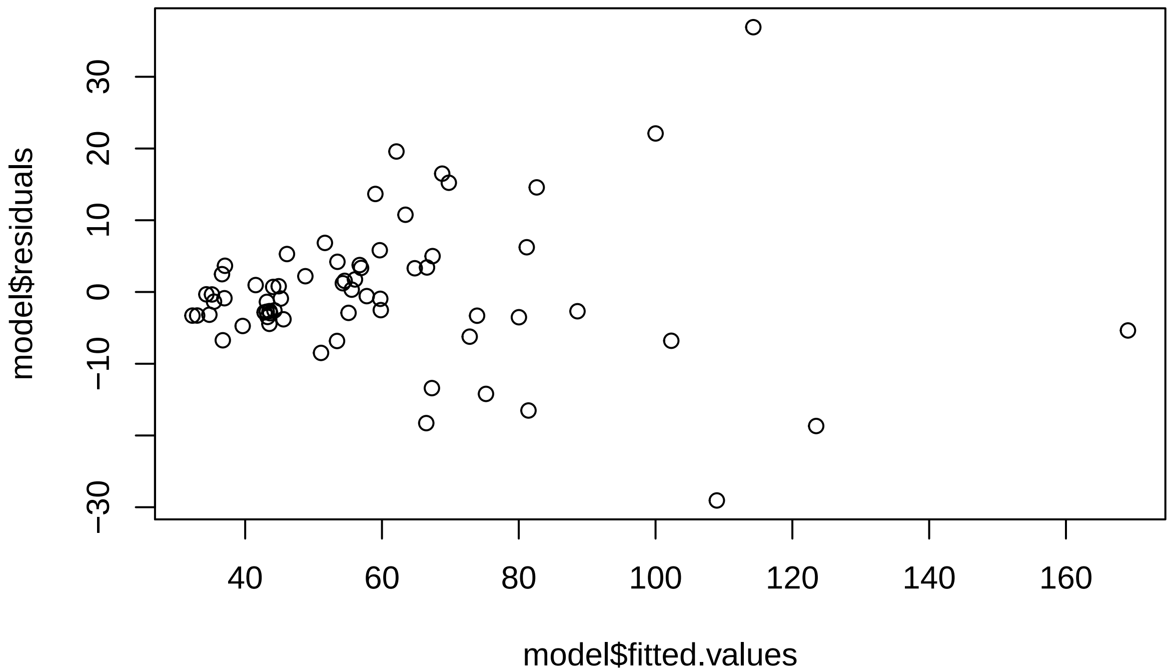
(a) Assign X2 and X2 based on 'Degree'

```
# Loop to assign X1 and X2
for (i in 1:nrow(df)) {
  if (df$degree[i]==1){
    df$x1[i] <- 0
    df$x2[i] <- 0
  }
  if (df$degree[i]==2){
    df$x1[i] <- 1
    df$x2[i] <- 0
  }
  if (df$degree[i]==3){
  df$x1[i] <- 0
  df$x2[i] <- 1
  }
}
attach(df)
```

(b) Fit Model, get Regression Plot

```
# Generate model
model <- lm(y~x1+x2+x3+x4)

# Plot residuals
plot(model$fitted.values, model$residuals)
```



Analysis: We see that in general, as the 'fitted' values get larger, they also tend to differ more from their corresponding 'actual' values, thus the larger residuals.

(c) Groups, and Brown-Forsythe

```
# Add Fitted values to df
df$fitted <- model$fitted.values

# Order dataframe by ascending 'Fitted' values
df <- df[order(df$fitted),]

# Make two groups (first 33 in 'gr1', and other 32 in 'gr2')
gr1 <- df[1:33,]
gr2 <- df[34:nrow(df),]

# DO BROWN FORSYTHE using the 2 groups
d1 <- abs((gr1$fitted-gr1$y)-median(gr1$fitted-gr1$y))  # Eqn 3.8
d2 <- abs((gr2$fitted-gr2$y)-median(gr2$fitted-gr2$y))

# Calculate std. dev from variance
s <- 0
for (k in 1:nrow(gr1)) {
  diff.1 <- (d1[k]-mean(d1))^2
  s <- s+diff.1
}
for (l in 1:nrow(gr2)){
  diff.2 <- (d2[l]-mean(d2))^2
  s <- s+diff.2
}
s <- s/(n-2)
s <- sqrt(s)
```

```r
# Calculate t-statistic for Brown-Forsythe
t.bf <- (mean(d1)-mean(d2)/(s*sqrt((1/nrow(gr1))+(1/nrow(gr2)))))
t.bf
```

```
## [1] -3.647595
```

```r
# Calculate critical t-statistic at alpha=0.01
t.crit <- qt(1-0.01,n-2)
```

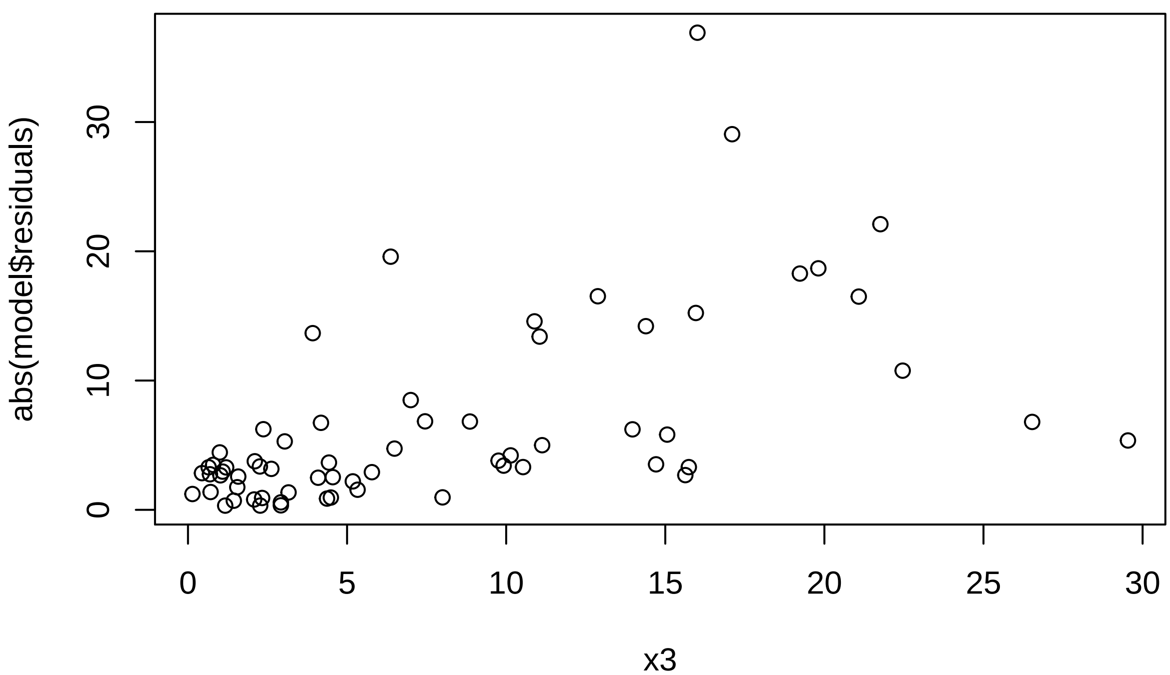
Decision Rules:
If the absolute value of 't.bf' <= 't.crit', conclude error variance is constant.
If the absolute value of 't.bf' > 't.crit', conclude error variance is not constant.

Conclusion:
t.bf = 3.6475953
t.crit = 2.3870079

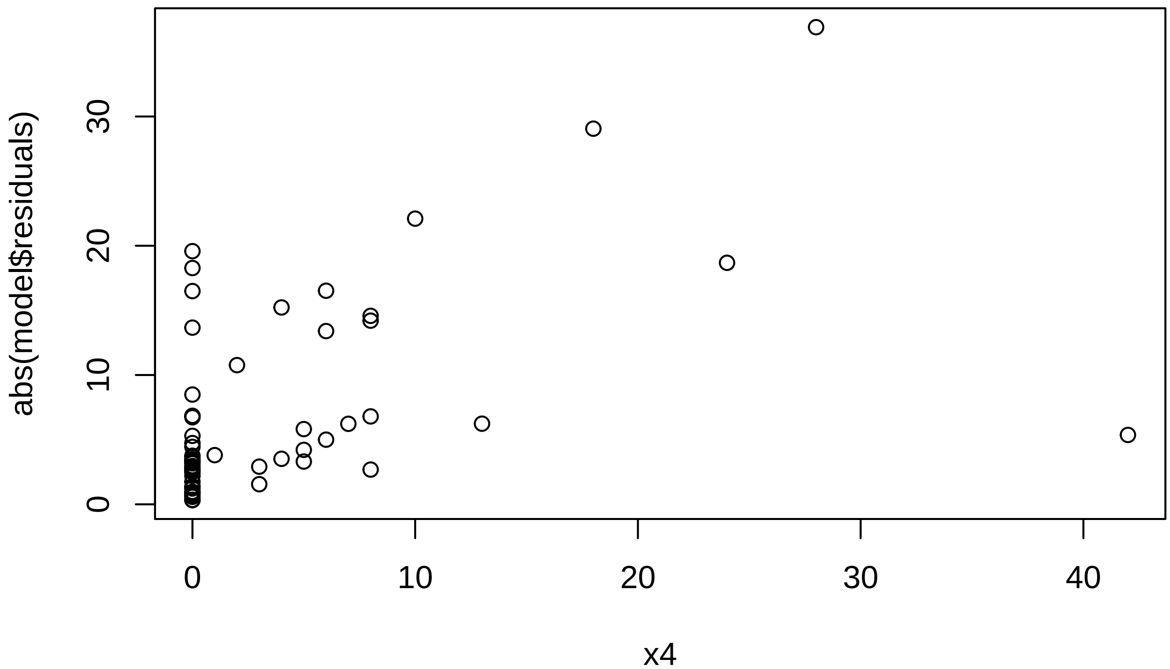
As 't.bf' > 't.crit', we conclude that error variance is NOT constant between Group 1 and Group 2.

(d) Plot absolute residuals with X3 and X4

```r
plot(x3,abs(model$residuals))
```




```r
plot(x4,abs(model$residuals))
```



What do these plots suggest about the relation between the stddev of the error term to X3 and X4?

Analysis: We see that there may be a linear relationship between the standard deviation of the error term and X3, due to the vaguely linear pattern we can visually identify. With X4, however, such a relationship is unclear.

(e) Estimate standard deviation fn, calculate estimated weights

```
# Capturing residuals
ei <- model$residuals
abs.ei <- abs(ei)

# Generating standard deviation functions against X3 and X4
model.1 <- lm(abs.ei~df$x3+df$x4)
s.1 <- model.1$fitted.values

# Below are the estimated weights for each case in the model:
wi <- 1/(s.1^2)
print(wi)
```

```
##           1           2           3           4           5           6
## 0.031287261 0.030720464 0.029661029 0.029326360 0.029043988 0.028821016
##           7           8           9          10          11          12
## 0.027980811 0.027901457 0.027735035 0.027682790 0.025974906 0.024821442
##          13          14          15          16          17          18
## 0.031507819 0.031245512 0.031224670 0.031141506 0.030924777 0.030904254
##          19          20          21          22          23          24
## 0.030832585 0.030477951 0.030337809 0.029845112 0.029603252 0.023184894
##          25          26          27          28          29          30
## 0.028932180 0.027041996 0.025576494 0.025232517 0.024198522 0.021502162
##          31          32          33          34          35          36
## 0.031826970 0.025539151 0.025195924 0.030750982 0.030367758 0.029825654
```

```
##          37          38          39          40          41          42
## 0.029670676 0.029043988 0.028131629 0.018847739 0.027630693 0.027578743
##          43          44          45          46          47          48
## 0.026070003 0.016351328 0.021271887 0.018225624 0.023462477 0.020645996
##          49          50          51          52          53          54
## 0.020601266 0.017395757 0.018688920 0.018820605 0.019961667 0.018345855
##          55          56          57          58          59          60
## 0.019306733 0.023532886 0.019658058 0.020104892 0.017777711 0.014920205
##          61          62          63          64          65
## 0.013747247 0.014946927 0.013466772 0.013032440 0.008814424
```

The above table shows the individual weights for each case

(f) Obtain weighted least squares fit

```r
model.w <- lm(y~df$x1+df$x2+df$x3+df$x4, weights=wi)

model
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4)
##
## Coefficients:
## (Intercept)           x1           x2           x3           x4
##      31.471       10.812       22.631        1.258        1.852
```

```r
model.w
```

```
##
## Call:
## lm(formula = y ~ df$x1 + df$x2 + df$x3 + df$x4, weights = wi)
##
## Coefficients:
## (Intercept)        df$x1        df$x2        df$x3        df$x4
##     64.2668      -5.7681      -0.8823      -0.1978      -0.0356
```

Are the WLS estimates of regression coefficients similar to the ones obtained with OLS?

Analysis: No, the coefficients are significantly different.

(g) Compare estimated standard deviation of the WLS coefficients in (f) with the OLS ones in (b)

```r
# The t-value of the coefficients, called by 'summary(model)', is an estimate
# of the standard deviation of the estimate.

summary(model)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -29.058  -3.477  -0.915   3.417  36.909
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.4714     2.8691  10.969 5.73e-16 ***
## x1           10.8120     3.2183   3.360  0.00136 **
## x2           22.6307     3.4846   6.494 1.81e-08 ***
## x3            1.2581     0.2273   5.535 7.23e-07 ***
## x4            1.8523     0.2276   8.137 2.86e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.14 on 60 degrees of freedom
## Multiple R-squared:  0.8633, Adjusted R-squared:  0.8542
## F-statistic: 94.76 on 4 and 60 DF,  p-value: < 2.2e-16
```

```r
summary(model.w)
```

```
##
## Call:
## lm(formula = y ~ df$x1 + df$x2 + df$x3 + df$x4, weights = wi)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6358 -2.3818 -0.9021  1.2617 17.2021
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  64.2668     7.6185   8.436 8.88e-12 ***
## df$x1        -5.7681     8.6041  -0.670    0.505
## df$x2        -0.8823     9.4207  -0.094    0.926
## df$x3        -0.1978     0.6953  -0.285    0.777
## df$x4        -0.0356     0.8183  -0.044    0.965
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.372 on 60 degrees of freedom
## Multiple R-squared:  0.01155,    Adjusted R-squared:  -0.05434
## F-statistic: 0.1753 on 4 and 60 DF,  p-value: 0.9502
```

What do you find?

Analysis: We find that the standard deviation estimates of the coefficients in the two models are precisely the same.

(h) Iterate steps (e)-(f) one more time

```r
# Generate 2nd standard deviation function, weights, and model
model.2 <- lm(abs(model.w$residuals)~df$x3 + df$x4)
s.2 <- model.2$fitted.values

# Below are the estimated weights for each case in the model:
wi.2 <- 1/(s.2^2)
```

```r
print(wi.2)
```

```
##           1           2           3           4           5           6 
## 0.002817677 0.002834639 0.002868079 0.002879146 0.002888683 0.002896347 
##           7           8           9          10          11          12 
## 0.002926333 0.002929260 0.002935453 0.002937413 0.003005905 0.003057644 
##          13          14          15          16          17          18 
## 0.002811241 0.002818906 0.002819520 0.002821981 0.002828454 0.002829071 
##          19          20          21          22          23          24 
## 0.002831235 0.002842089 0.002846448 0.002862098 0.002869972 0.003031391 
##          25          26          27          28          29          30 
## 0.002892511 0.002962076 0.003023232 0.003038645 0.003087710 0.002724373 
##          31          32          33          34          35          36 
## 0.002802085 0.002731178 0.002744410 0.002833710 0.002845513 0.002862727 
##          37          38          39          40          41          42 
## 0.002867764 0.002888683 0.002920817 0.002873865 0.002939374 0.002941338 
##          43          44          45          46          47          48 
## 0.003001850 0.003412674 0.002735793 0.003488081 0.003125383 0.002677942 
##          49          50          51          52          53          54 
## 0.002680219 0.003567645 0.002985195 0.002689318 0.003345058 0.002630157 
##          55          56          57          58          59          60 
## 0.002943782 0.002061173 0.002730766 0.002535863 0.002664829 0.002695798 
##          61          62          63          64          65 
## 0.003001485 0.002102570 0.001664880 0.001876798 0.001398314
```

```r
model.w2 <- lm(y~df$x1+df$x2+df$x3+df$x4, weights=wi.2)

# Compare model coefficients
model.w
```

```
## 
## Call:
## lm(formula = y ~ df$x1 + df$x2 + df$x3 + df$x4, weights = wi)
## 
## Coefficients:
## (Intercept)        df$x1        df$x2        df$x3        df$x4  
##     64.2668      -5.7681      -0.8823      -0.1978      -0.0356
```

```r
model.w2
```

```
## 
## Call:
## lm(formula = y ~ df$x1 + df$x2 + df$x3 + df$x4, weights = wi.2)
## 
## Coefficients:
## (Intercept)        df$x1        df$x2        df$x3        df$x4  
##     63.7305      -7.1578       0.4476      -0.1653       0.1176
```

Is there substantial change in the coefficients?

Analysis: Yes, there is substantial change, especially for X4.

Problem 11.23 - Cement Composition

```r
# Import data
rm(list=ls())
colnames <- c("y","x1","x2","x3","x4")
df <- read.table(url("http://users.stat.ufl.edu/~rrandles/sta4210/Rclassnotes/data/textdatasets/KutnerDa
n <- nrow(df)
attach(df)
```

(a) Fit regression, state function

```r
model <- lm(y~x1+x2+x3+x4)
model
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4)
##
## Coefficients:
## (Intercept)           x1           x2           x3           x4
##     62.4054       1.5511       0.5102       0.1019      -0.1441
```

Model: 62.4054 + 1.5511*X1 + 0.5102*X2 + 0.1019*X3 - 0.1441*X4

(b) Obtain estimated ridge standardized regression coefficients, VIF, R^2

```r
lambda1 <- c(.000,.002,.004,.006,.008,.02,.04,.06,.08,.1)
model.r <- ridge(y,as.matrix(df[,2:5]),lambda=lambda1)

# Coefficients of ridge
coef(model.r)
```

```
##             x1       x2         x3        x4
## 0.000 8.766245 7.627215  0.6271348 -2.316721
## 0.002 8.646498 7.316466  0.4947726 -2.644141
## 0.004 8.545395 7.055341  0.3831741 -2.919231
## 0.006 8.458811 6.832856  0.2877444 -3.153575
## 0.008 8.383753 6.641046  0.2051516 -3.355571
## 0.020 8.081286 5.883629 -0.1257230 -4.152646
## 0.040 7.827261 5.286090 -0.3987136 -4.779991
## 0.060 7.685670 4.986685 -0.5465554 -5.092887
## 0.080 7.590013 4.808397 -0.6433106 -5.277999
## 0.100 7.517712 4.691142 -0.7141240 -5.398689
```

```r
# Variance inflation factors
vif(model.r)
```

```
##              x1        x2        x3         x4
## 0.000 38.496211 254.42317 46.868386 282.512865
## 0.002 32.575987 212.48455 39.526169 235.878725
## 0.004 28.011884 180.15940 33.866070 199.934766
## 0.006 24.418985 154.71882 29.410644 171.646398
## 0.008 21.539859 134.33773 25.840571 148.984139
```

```
## 0.020 12.144405  67.89764 14.193439  75.110465
## 0.040  6.946266  31.29338  7.756845  34.415703
## 0.060  5.067162  18.19643  5.436762  19.858527
## 0.080  4.171867  12.05997  4.336651  13.039955
## 0.100  3.669694   8.69994  3.723859   9.307785
```

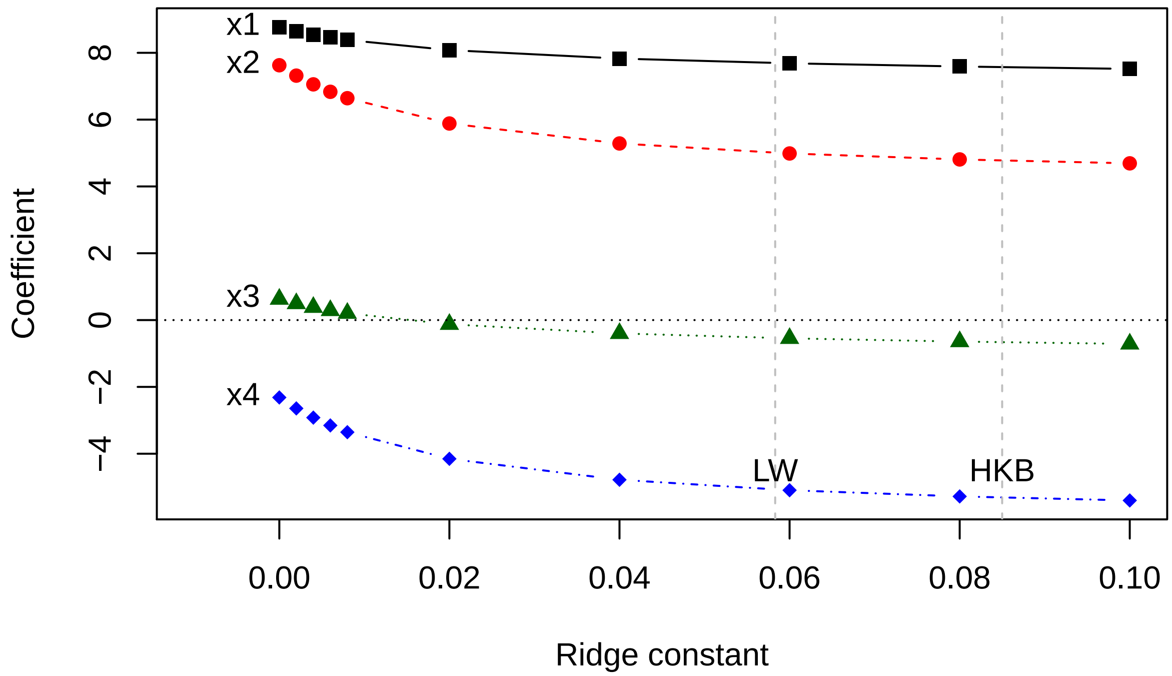
```
# R^2
## Using 'lmridge' package to calculate R^2 values ##

model.r2 <- lmridge(y~x1+x2+x3+x4, data=df, K=lambda1)
rstats1(model.r2) # provides R2 and adj-R2 values
```

```
##
## Ridge Regression Statistics 1:
##
##          Variance   Bias^2       MSE rsigma2        F     R2 adj-R2
## K=0     3309.5049   0.0000 3309.5049  5.3182 125.4142 0.9824 0.9765
## K=0.002  671.5366 124.5895  796.1261  5.1666 129.0947 0.9801 0.9735
## K=0.004  291.9900 206.9868  498.9768  5.0733 131.4684 0.9780 0.9707
## K=0.006  171.0478 253.5305  424.5783  5.0246 132.7411 0.9759 0.9679
## K=0.008  117.5860 283.0067  400.5927  4.9970 133.4760 0.9739 0.9652
## K=0.02    44.2840 352.2366  396.5206  4.9745 134.0801 0.9619 0.9493
## K=0.04    28.5701 385.7555  414.3256  5.0960 130.8816 0.9430 0.9240
## K=0.06    23.8909 403.1903  427.0812  5.2984 125.8813 0.9250 0.9000
## K=0.08    21.4311 416.6257  438.0568  5.5512 120.1503 0.9079 0.8772
## K=0.1     19.8579 428.4112  448.2692  5.8409 114.1900 0.8914 0.8552
##                CN
## K=0     1376.8806
## K=0.002  617.5113
## K=0.004  398.2584
## K=0.006  294.0423
## K=0.008  233.1425
## K=0.02   104.3161
## K=0.04    54.6732
## K=0.06    37.2536
## K=0.08    28.3705
## K=0.1     22.9838
```

(c) Make ridge trace plot

```
traceplot(model.r)
```



Do the ridge regression coedfficients exhibit substantial changes near c=0?

Analysis: Around c=0, the ridge coefficients DO exhibit substantial changes.

(d) Suggest reasonable 'c' based on ridge trace, VIF, and R-squared values

Response: I would choose **c = 0.1** as the most appropriate. This is the 'c' value at which the VIF values are most near 1, and where the ridge trace lines stabilize the most. The trade-off is in the R-squared values, as c=0.1 has the lowest R-squared value. However, and adj-R2 = 0.8552, it is still a good value overall.

(e) Transform ridge regression coefficients back to original variables

```r
# Using formulas 7.43c-d and 7.53a-b
## Calculating necessary values

## s values according to 7.43c-d
syy <- 0
for (i in 1:nrow(df)) {
  y.temp <- (y[i] - mean(y))^2
  syy <- syy + y.temp
}

sx1 <- 0
for (i in 1:nrow(df)) {
  x1.temp <- (x1[i] - mean(x1))^2
  sx1 <- sx1 + x1.temp
}
sx2 <- 0
for (i in 1:nrow(df)) {
  x2.temp <- (x2[i] - mean(x2))^2
  sx2 <- sx2 + x2.temp
```

```r
}
sx3 <- 0
for (i in 1:nrow(df)) {
  x3.temp <- (x3[i] - mean(x3))^2
  sx3 <- sx3 + x3.temp
}
sx4 <- 0
for (i in 1:nrow(df)) {
  x4.temp <- (x4[i] - mean(x4))^2
  sx4 <- sx4 + x4.temp
}

sy <- sqrt(syy/(n-1))
s1 <- sqrt(sx1/(n-1))
s2 <- sqrt(sx2/(n-1))
s3 <- sqrt(sx3/(n-1))
s4 <- sqrt(sx4/(n-1))

# Calculating new TRANSFORMED coefficients using 7.53a-b
b1.new <- (sy/s1)*model.r$coef[1]
b2.new <- (sy/s2)*model.r$coef[2]
b3.new <- (sy/s3)*model.r$coef[3]
b4.new <- (sy/s4)*model.r$coef[4]

b0.new <- mean(y) - b1.new*mean(x1) - b2.new*mean(x2) - b3.new*mean(x3) - b4.new*mean(x4)   #7.53b

# Creating results vector
new.fits <- as.vector(c(rep(0,13)))

# Looping through 'df' to calculate new fits and saving in 'new.fits'
for (j in 1:nrow(df)) {
  fit <- b0.new + b1.new*x1[j] + b2.new*x2[j] + b3.new*x3[j] + b4.new*x4[j]
  new.fits[j] <- fit
}

# Compare transformed fits to original fits
new.fits
```

```
##  [1]  12.17208  22.55110  88.66303  84.95212  24.24213 115.57947 108.89884
##  [8] 118.94339  86.08516 202.95277 138.22120 131.50497 105.73374
```

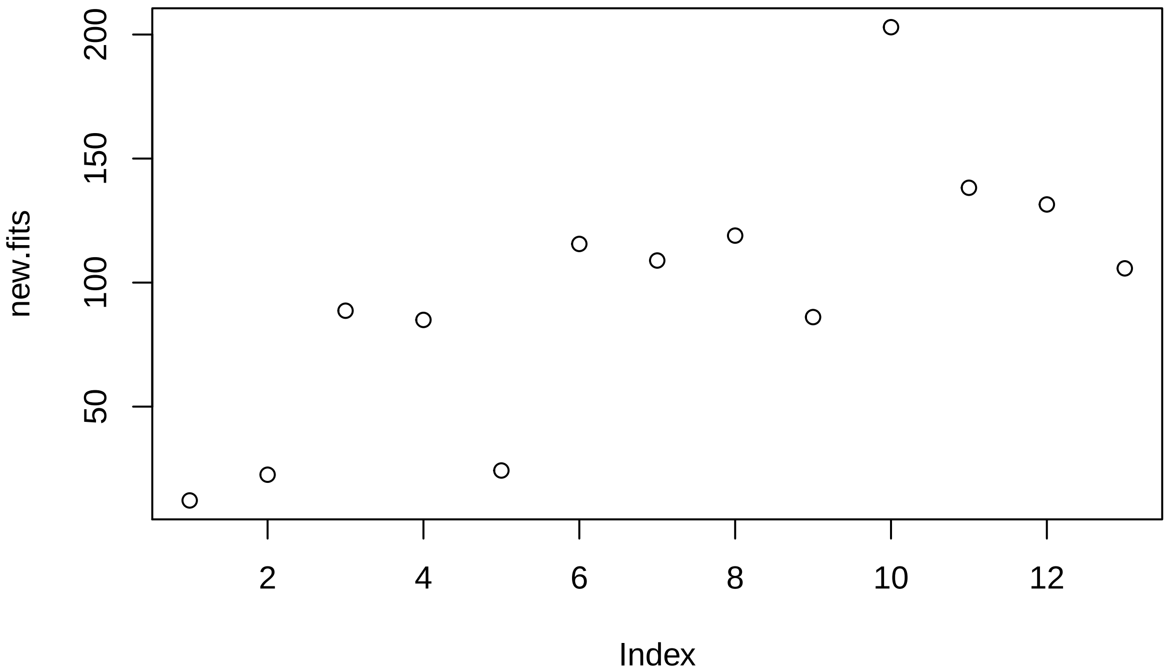
```r
model$fitted.values
```

```
##         1         2         3         4         5         6         7
##  78.49524  72.78880 105.97094  89.32710  95.64924 105.27456 104.14867
##         8         9        10        11        12        13
##  75.67499  91.72165 115.61845  81.80902 112.32701 111.69433
```

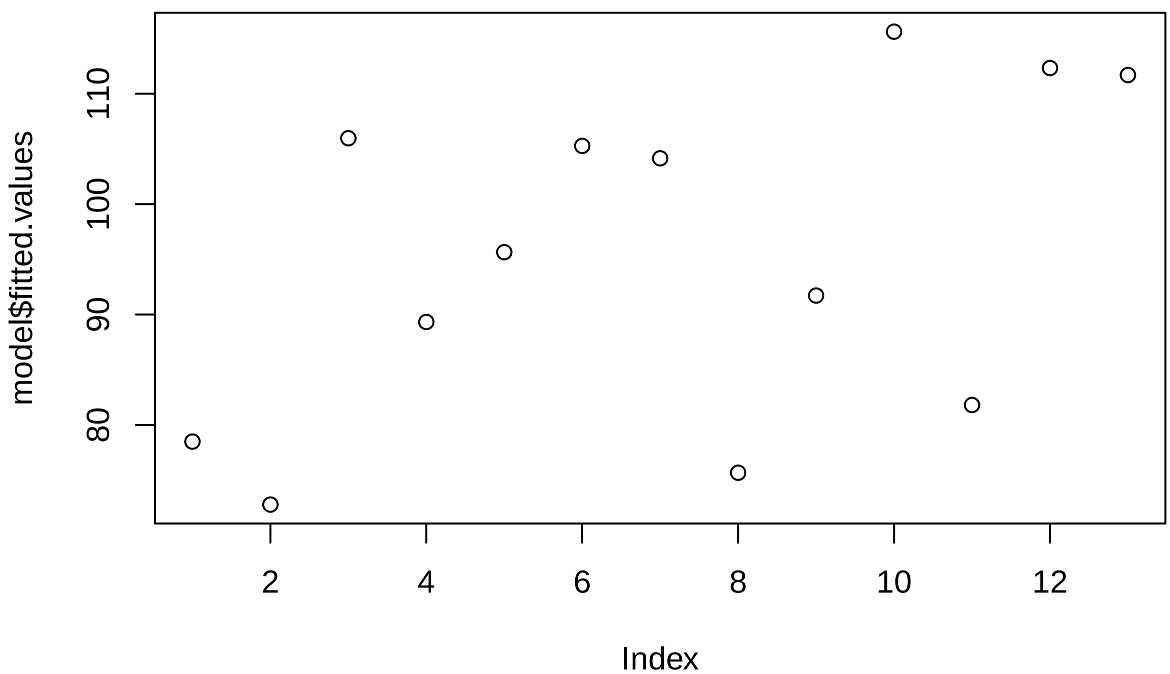
```r
mean(abs(new.fits-model$fitted.values))
```

```
## [1] 34.03814
```

```r
plot(new.fits)
```




```r
plot(model$fitted.values)
```



Analysis: After transforming the ridge regression coefficients back onto the original variables, we can analyze the difference in fitted values. Numerically, the mean absolute difference is 34.0381445, while visually when we plot the two sets, we can notice that the original fitted values reach a max of about 115, while the new fitted values climb has high as 150, and even 200. The patterns of increase differ as well.

Problem 11.28 - Mileage Study

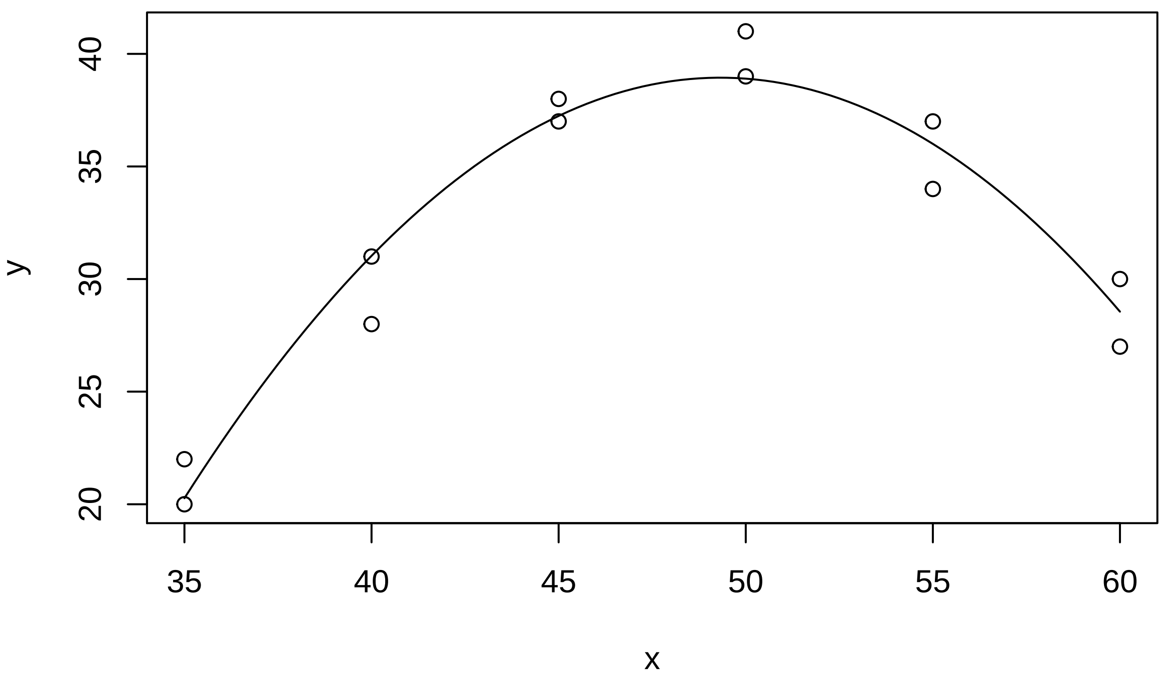
```
rm(list=ls())
colnames <- c("y","x")
df <- read.table(url("http://users.stat.ufl.edu/~rrandles/sta4210/Rclassnotes/data/textdatasets/KutnerDa
n <- nrow(df)
attach(df)
```

(a) Fit 2nd-order regression model

```
model <- lm(y~poly(x,2))
model
```

```
##
## Call:
## lm(formula = y ~ poly(x, 2))
##
## Coefficients:
## (Intercept)  poly(x, 2)1  poly(x, 2)2
##      32.000        9.804      -19.674
```

```
plot(x,y)
curve(predict(model,newdata=data.frame(x=x)),add=T)
```



Analysis: Looking at the plot, YES the 2nd-order function looks to be a good fit.

(b) Estimate Xmax

```
x.hat.max <- mean(x) - (0.5*9.804/(-19.674))
y.hat <- 32.000 + x.hat.max*9.804 + (x.hat.max^2)*(-19.674)
```

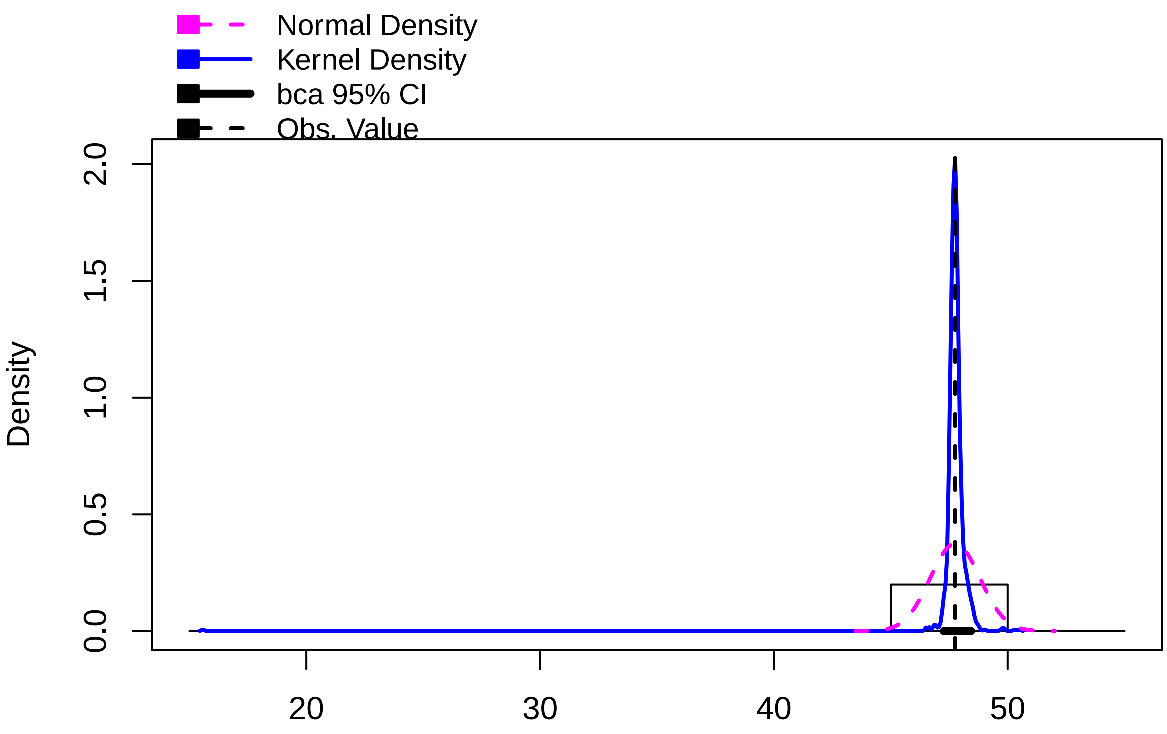
Results: The estimated speed Xmax is 47.7491613 and the mean milage at that speed is -4.4356241×10^4.

(c) Bootstrap sampling

```r
# Using 'boot' package and 'boot()' function to create bootstrap sampling

# Defining function to be passed into 'boot()' function (returning the X.max.hat)
xmaxfn <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  x.hat.max.boot <- mean(x) - (0.5*fit$coefficients[2]/(fit$coefficients[3]))
  return(x.hat.max.boot)
}
# bootstrapping with 1000 replications
results <- boot(data=df, statistic=xmaxfn,
   R=1000, formula=y~poly(x,2))

# Histogram of bootstrap sample results
hist(results)
```



Analysis: We see from the histogram that YES, the results of the X-max-hat bootstrap sampling appear to be Normal.

Problem 11.30 - Patient Satisfaction

```r
rm(list=ls())
colnames <- c("y","x1","x2","x3")
df <- read.table(url("http://users.stat.ufl.edu/~rrandles/sta4210/Rclassnotes/data/textdatasets/KutnerDa
```

```r
n <- nrow(df)
attach(df)
```

Fit two-region regression tree

```r
tree <- rpart(y~x1+x2) # By default, the tree created only had 2 splits, and since the book questions a
tree <- rpart(y~x1+x2,control=list(minsplit=1,cp=.000001))  # I Manipulated the cp value to generate mo
tree
```

```
## n= 46
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 46 13369.3000 61.56522
##    2) x1>=36.5 24  3513.8330 50.08333
##      4) x1>=46 10   713.6000 40.80000 *
##      5) x1< 46 14  1322.8570 56.71429
##       10) x1< 42.5 7   567.4286 55.71429 *
##       11) x1>=42.5 7   741.4286 57.71429 *
##    3) x1< 36.5 22  3239.8180 74.09091
##      6) x1>=29.5 12  1254.0000 67.00000 *
##      7) x1< 29.5 10   658.4000 82.60000 *
```

(a) First split point is based on X1, at X1 = 36.5, SSE = 13369.3000

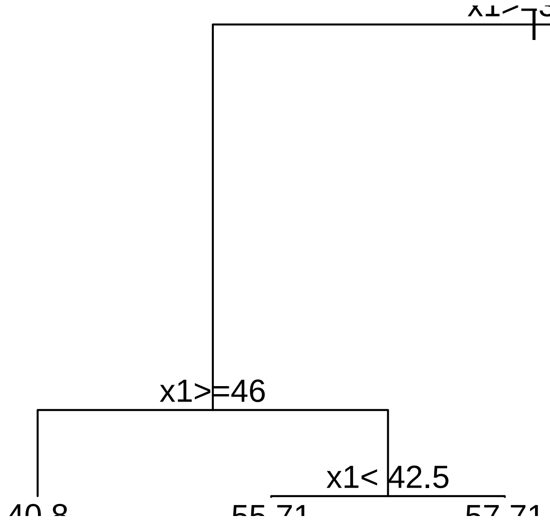
(b) Second split point is based on X1, at X1 = 46.10, SSE = 713.600

(c) Third split point is based on X1, at X1 = 29.5, SSE = 658.400

(d) Fourth split point is based on X1, at X1 = 42.5, SSE = 567.4286

(e) Plot tree

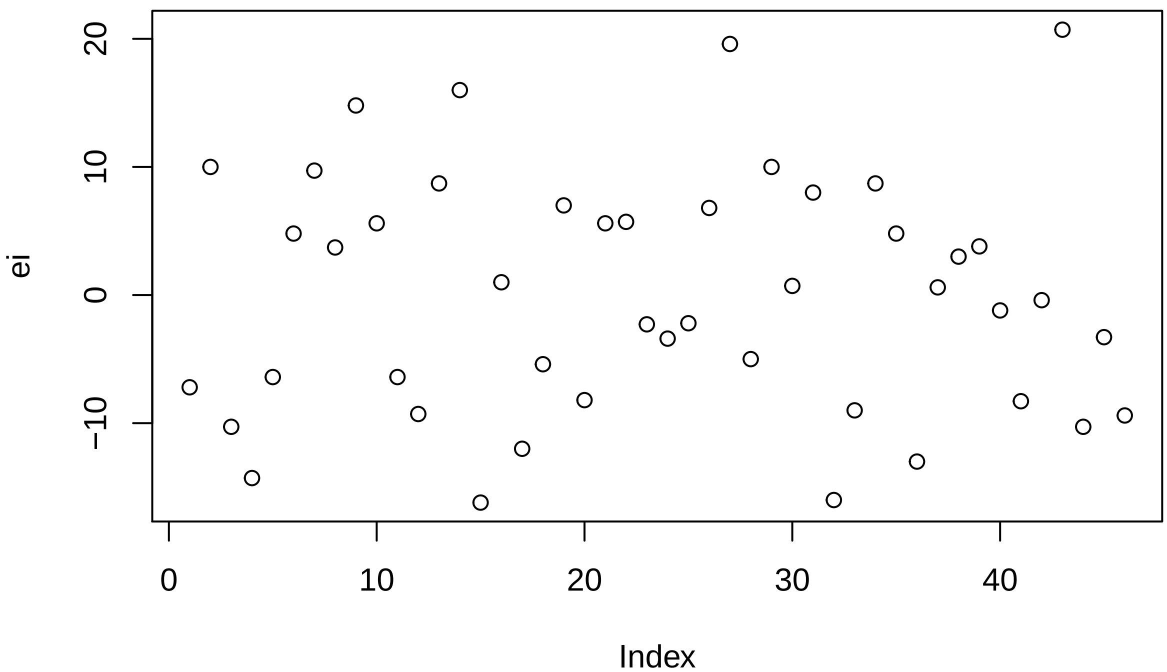
```r
plot(tree)
text(tree)
```



Analysis: Judging by this tree, we would say that X1 appears to be relatively more important than X2.

(f) Residual Plot

```r
# Generate fitted values of Tree
pred <- predict(tree)
ei <- pred-y
plot(ei)
```



Analysis: We see that the residuals center around 0, but have relatively dramatic swings both up and down (a residual of 20 is large for the context of this 'y' value).

Problem 11.31 - Prostate cancer

```
# Import data
rm(list=ls())
colnames <- c("id","y","x1","x2","x3","x4","x5","x6","x7")
df <- read.table(url("http://users.stat.ufl.edu/~rrandles/sta4210/Rclassnotes/data/textdatasets/KutnerDa
df$id <- NULL # Take out 'id' column as it's not a predictor
n <- nrow(df)
attach(df)
```

(a) Sample and Fit Tree

```
set.seed(5)
train.rows <- sample(nrow(df), 65,replace=FALSE)
train <- df[train.rows,]
test <- df[-train.rows,]

tree <- rpart(y ~ x1+x2+x3+x4+x5+x6+x7,data=train)
tree
```

```
## n= 65
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 65 94867.520  23.591290
##   2) x1< 15.9714 58  9917.298  14.281910
##     4) x5< 0.5 49  4468.193  11.365120
##       8) x1< 6.7341 39  1719.213   8.929692 *
##       9) x1>=6.7341 10  1615.505  20.863300 *
##     5) x5>=0.5 9  2762.572  30.162220 *
##   3) x1>=15.9714 7 38275.180 100.726100 *
```

Justification: The default rpart() tree is based on recommended control considerations (i.e. ratios between 'minsplit','cp','maxdepth',etc.) thus is more or less acceptable for use here.

(b) Evaluate predictive capability

```
# Using 'test' or remainder of the observations to evaluate predictive capability
pred <- predict(tree, newdata=test)

# Calculate mean squared error as a metric, and also mae as a % of the PSA value
mae <- mean(abs(pred-test$y))
mae.percentage <- mae/test$y*100
mae.percentage
```

```
##  [1] 2489.385642 1464.749010  742.112165  595.607011  595.607011
##  [6]  523.047242  411.436773  387.460096  375.989464  361.259848
## [11]  237.376225  216.933269  170.645793  145.420402  145.420402
## [16]  139.720459  130.271885  127.691545  123.916602  122.683744
## [21]  119.061220  114.392782  108.811644  104.547571   79.807216
## [26]   75.160586   71.494525   63.410565   59.716659   58.534983
```

```
## [31]   37.698522    8.001436
```

Analysis: We see that the 'mae' value is 21.2095657, which for the majority of the observations, is substatial relative to 'y' or 'PSA' value. This is supported by the vector 'mae.percentage', which prints out the percentage error for each observaion in the test set. We see that it regularly is extremely high - in fact, only in 1 case is it acceptable 8.0014357 %.

Thus, I would conclude this model would be very poor for predictive use by doctors.

(c) Compare with previous model

Case study 9.30 was never assigned. Thus, I will compare the above Tree model with a simple OLS regression.

```
model.comp <- lm(y~.,data=train)
pred.comp <- predict(model.comp, newdata=test)
mae.comp <- mean(abs(pred.comp-test$y))
mae.percentage.comp <- mae.comp/test$y*100
mae.percentage.comp
```

```
##  [1] 1743.118807 1025.647254  519.642136  417.056227  417.056227
##  [6]  366.248391  288.096455  271.307494  263.275523  252.961544
## [11]  166.215694  151.901117  119.489679  101.826344  101.826344
## [16]   97.835127   91.219042   89.412235   86.768943   85.905670
## [21]   83.369104   80.100168   76.192142   73.206350   55.882647
## [26]   52.628981   50.061930   44.401376   41.814827   40.987394
## [31]   26.397277    5.602769
```

Comparison: We see that the tree model is slightly worse than the standard OLS regression model, although it is quite slight. The OLS model has a slightly better MAE, but as we see from the error percentages, they are both very (very) high for many cases.