3.7 Exercises

- Exercise 8
- Exercise 9
- Exercise 10
- Exercise 11
- Exercise 12
- Exercise 13
- Exercise 14
- Exercise 15

Exercise 8

This question involves the use of simple linear regression on the **Auto** data set.

```
fh = 'D:/GoogleDrive/Introduction to Statistical Learning with Applications in R/data-se
ts/Auto.csv'
Auto = read.csv(file=fh, header=T, na.strings='?')
fix(Auto)
```

a. Use the **lm()** function to perform a simple linear regression with **mpg** as the response and **horsepower** as the predictor. Use the **summary()** function to print the results.

```
lm.fit = lm(mpg~horsepower, data=Auto)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

i. Is there a relationship between the predictor and the response? *The p-value is very low, so we may reject the null hypothesis and assume that some relationship does exist between the miles per gallon (mpg) of an automobile and its horsepower.*

ii. How strong is the relationship between the predictor and the response? *The R^2 value is 0.6059, so about 60% of the variance in a car's mpg can be explained by the car's horsepower.*

iii. Is the relationship between the predictor and the response positive or negative? $\hat{\beta}_1 = -0.157845$*, so the relationship is negative (i.e. an increase in a car's horsepower leads to a decrease in a car's fuel efficiency).*

iv. What is the predicted **mpg** associated with a **horsepower** of 98? What are the associated 95% confidence and prediction intervals?

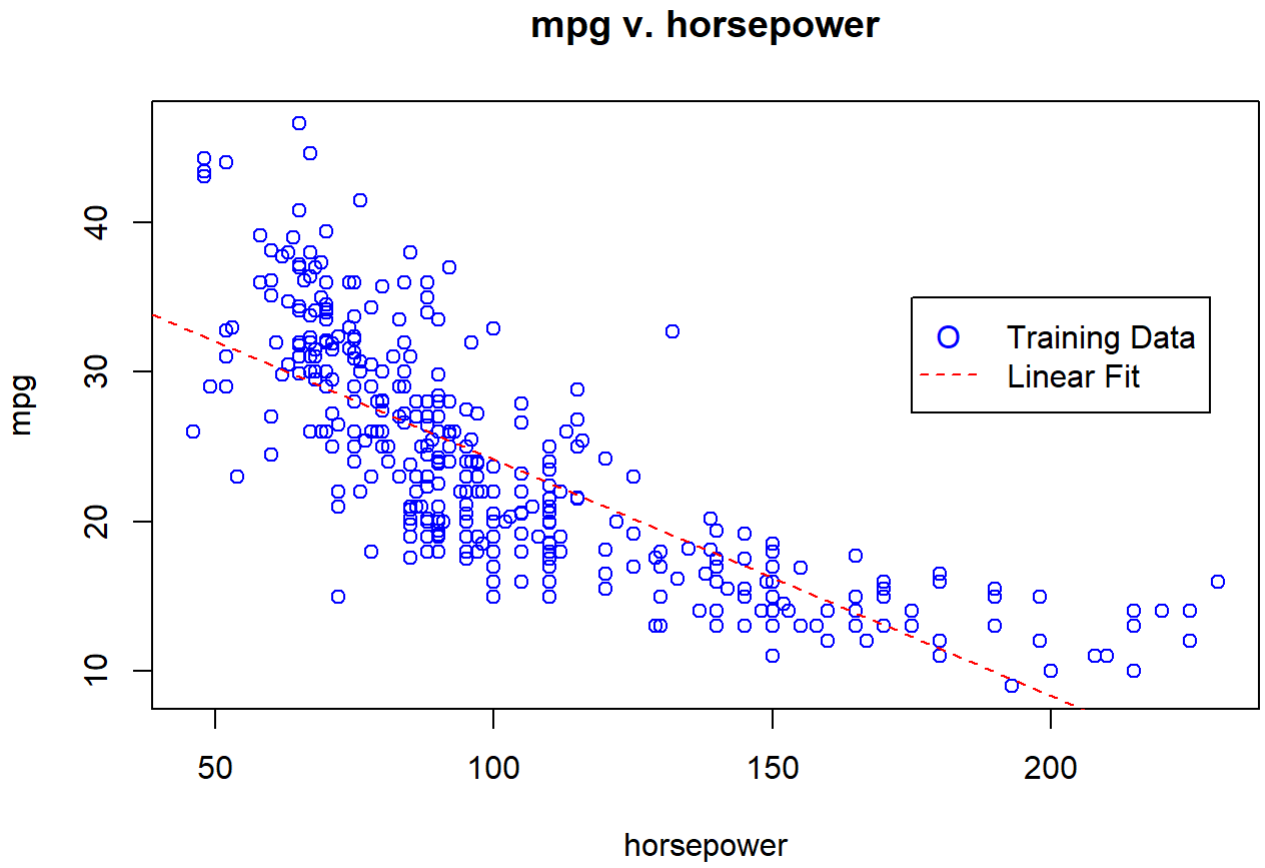
```
for (t in c('prediction', 'confidence')){
  print(t)
  print(predict(lm.fit, data.frame(horsepower = c(98)), interval = t, level = .95))
}
```

```
## [1] "prediction"
##        fit     lwr      upr
## 1 24.46708 14.8094 34.12476
## [1] "confidence"
##        fit      lwr      upr
## 1 24.46708 23.97308 24.96108
```

The predicted mpg *for a* horsepower *of 98 is 24.6708. The 95% prediction interval is bounded by [14.8094, 34.12476] and the 95% confidence interval is bounded by [23.97308, 24.96108]. Therefore, are 95% sure that the true mean mpg for cars with a horsepower of 98 is between 23.97 and 24.96. In general, we expect 95% of all cars with a horsepower of 98 to have an mpg between 14.81 and 34.12.*

b. Plot the response and the predictor. Use the **abline()** function to display the least squares regression line.

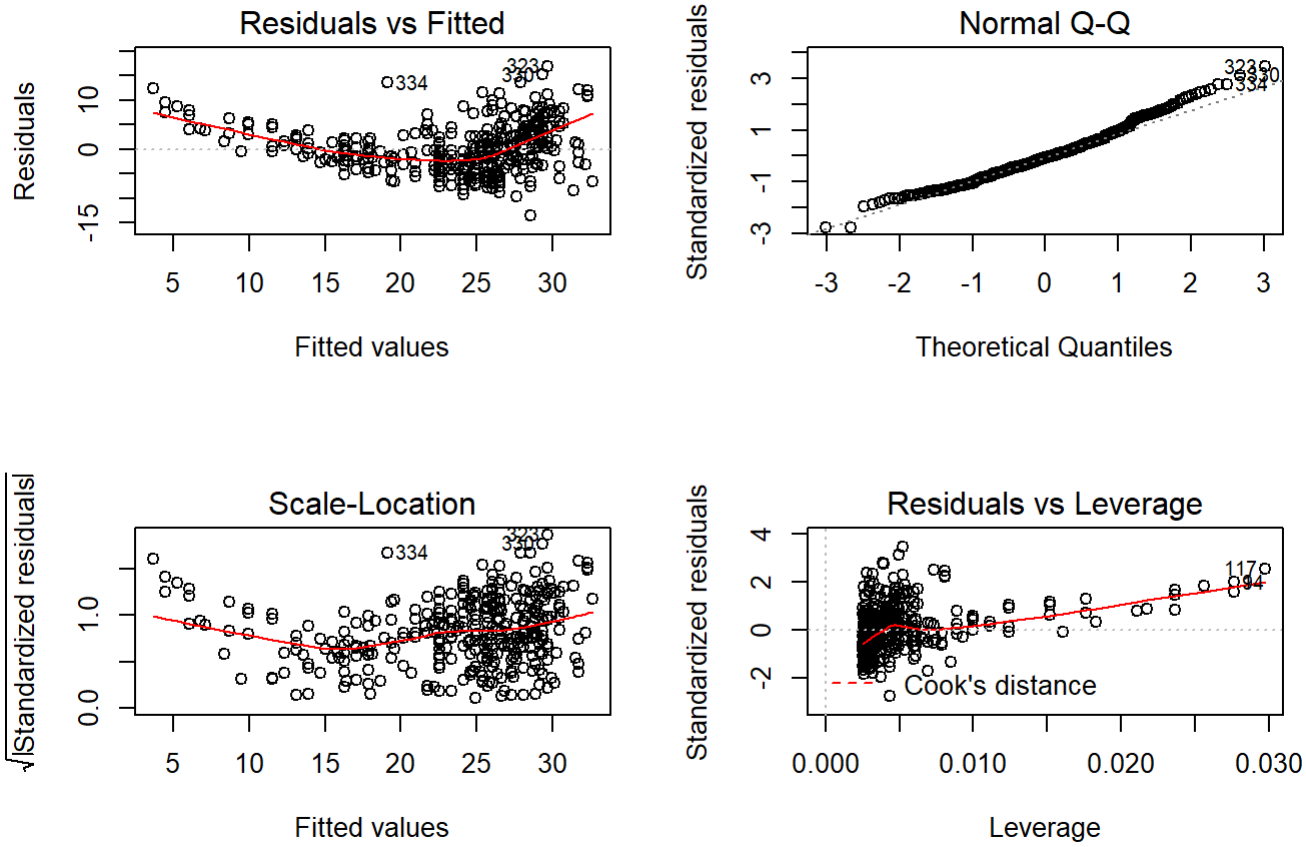
```
plot(x = Auto$horsepower, y = Auto$mpg, xlab='horsepower', ylab='mpg', main='mpg v. hors
epower', col = 'blue')
abline(lm.fit, col = 'red', lty=2)
legend(175, 35, legend = c('Training Data', 'Linear Fit'), lty = c(NA,2), pch = c('O', N
A), col = c('blue', 'red'))
```



From a first look at the plotted data, it seems like a quadratic function or an exponential decay might fit better.

c. Use the **plot()** function to produce diagnostic plots of the least squares regression fit. Comment on any problems you see with the fit.

```
par(mfrow=c(2,2))
plot(lm.fit)
```



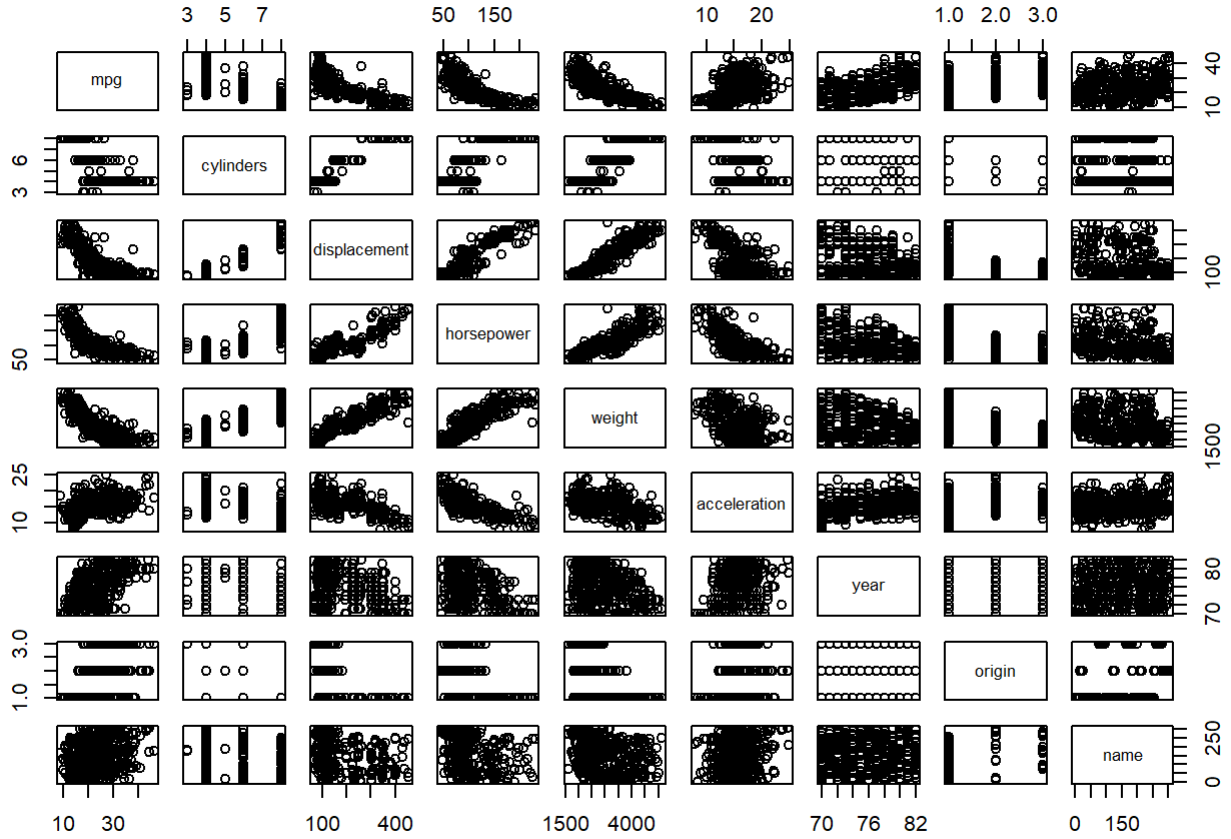
Looking at the plot of Residuals vs Fitted values, there is a clear U-shape in the residuals. This trend suggests that a linear fit is probably not the best method for estimating the relationship between mpg and horsepower.

Exercise 9

This question involves the use of multiple linear regression on the **Auto** data set.

a. Produce a scatterplot matrix which includes all of the variables in the data set.

```
pairs(Auto)
```



b. Compute the matrix of correlations between the variables using the function **cor()**. You will need to exclude the **name** variable, which is qualitative.

```
cor(Auto[,-which(names(Auto) == 'name')])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7762599   -0.8044430         NA -0.8317389
## cylinders    -0.7762599  1.0000000    0.9509199         NA  0.8970169
## displacement -0.8044430  0.9509199    1.0000000         NA  0.9331044
## horsepower           NA         NA           NA          1         NA
## weight       -0.8317389  0.8970169    0.9331044         NA  1.0000000
## acceleration  0.4222974 -0.5040606   -0.5441618         NA -0.4195023
## year          0.5814695 -0.3467172   -0.3698041         NA -0.3079004
## origin        0.5636979 -0.5649716   -0.6106643         NA -0.5812652
##              acceleration       year     origin
## mpg             0.4222974  0.5814695  0.5636979
## cylinders      -0.5040606 -0.3467172 -0.5649716
## displacement   -0.5441618 -0.3698041 -0.6106643
## horsepower             NA         NA         NA
## weight         -0.4195023 -0.3079004 -0.5812652
## acceleration    1.0000000  0.2829009  0.2100836
## year            0.2829009  1.0000000  0.1843141
## origin          0.2100836  0.1843141  1.0000000
```

The data pairs with high magnitudes of correlation also have scatter plots with patterns in the data.

c. Use the **lm()** function to perform a multiple linear regression with **mpg** as the response and all other variables except **name** as the predictors. Use the **summary()** function to print the results.

```
lm.fit = lm(mpg~.-name, data = Auto)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

i. Is there a relationship between the predictors and the response?

The F-statistic is quite large and the p-value is very small, so we may reject the null hypothesis and say that there is a relationship between the predictors and the mpg.

ii. Which predictors appear to have a statistically significant relationship to the response?

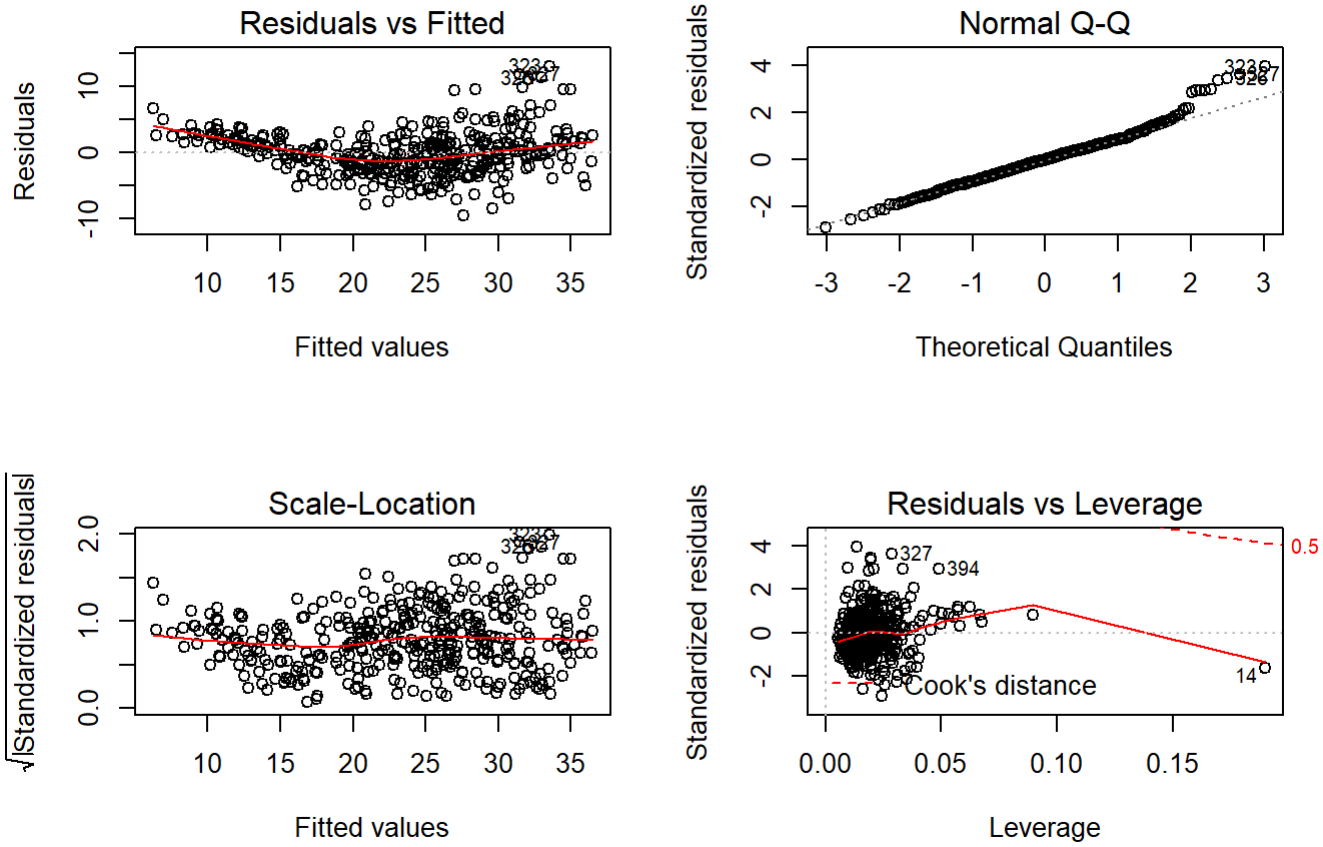
Displacement, weight, year, and origin all have p-values less than 1% (as does the Intercept). These predictors therefore seem to be statistically significant.

iii. What does the coefficient for the **year** variable suggest?

$\hat{\beta_{year}} = 0.750773$ *and has a very low p-value. The data therefore suggests that fuel efficiency has increased by about 0.75 miles per gallon each year (ceteris paribus).*

d. Use the **plot()** function to produce diagnostic plots of the linear regression fit. Comment on any probelms you see with the fit. Do the residual plots suggest any unusually large outliers? Does the leverage plot identify any observations with unusually high leverage?

```
par(mfrow=c(2,2))
plot(lm.fit)
```



There does still seem to be a trend in the residuals plot, but it does look like a better fit than the simple linear regression using just horsepower as a predictor.

The plots reveal the 327th and 394th observations to be outliers and the 14th observation to be a high leverage point.

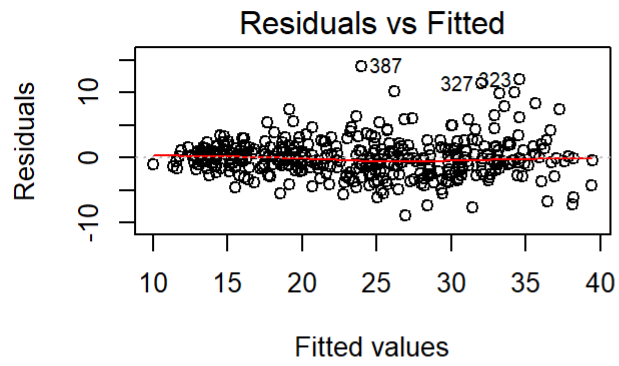
e. Use the ***** and **:** symbols to fit linear regression models with interaction effects. Do any interactions appear to be statistically significant?

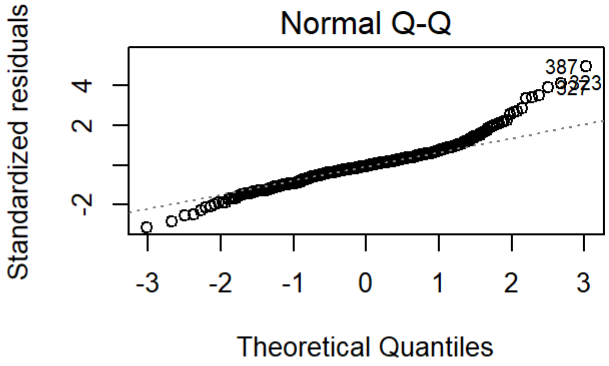
```
lm.fit = lm(mpg ~ displacement*weight + displacement*year + displacement*origin + weight
*year + weight*origin + year*origin, data = Auto)
summary(lm.fit)
```

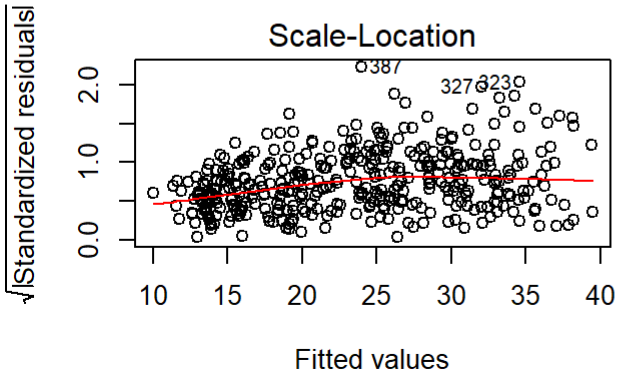
```
##
## Call:
## lm(formula = mpg ~ displacement * weight + displacement * year +
##     displacement * origin + weight * year + weight * origin +
##     year * origin, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9244 -1.5779 -0.1339  1.2274 14.0364
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -2.172e+01  2.449e+01  -0.887  0.37571
## displacement         2.377e-02  8.216e-02   0.289  0.77250
## weight              -6.472e-03  1.108e-02  -0.584  0.55947
## year                 9.586e-01  3.182e-01   3.012  0.00276 **
## origin              -5.630e+00  5.253e+00  -1.072  0.28453
## displacement:weight  1.814e-05  2.725e-06   6.657  9.6e-11 ***
## displacement:year   -1.446e-03  1.126e-03  -1.285  0.19971
## displacement:origin  1.545e-02  1.272e-02   1.215  0.22503
## weight:year         -2.067e-05  1.496e-04  -0.138  0.89019
## weight:origin       -1.275e-03  9.415e-04  -1.355  0.17628
## year:origin          9.324e-02  6.514e-02   1.431  0.15314
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.944 on 386 degrees of freedom
## Multiple R-squared:  0.8621, Adjusted R-squared:  0.8585
## F-statistic: 241.3 on 10 and 386 DF,  p-value: < 2.2e-16
```

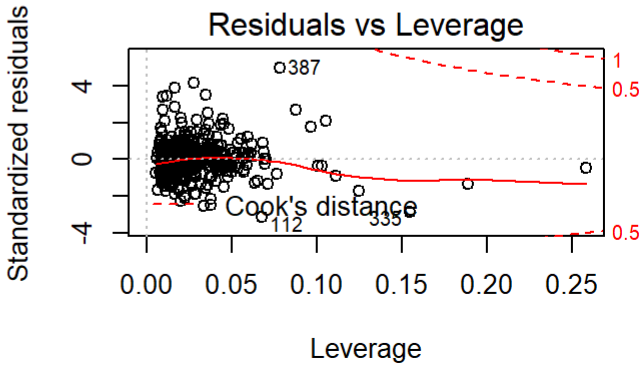
In this fit, I used all of the predictors that had low p-values in the original multiple linear regression. I included an interaction term pairwise between each of those predictors. Surprisingly, very few of the resulting terms have low associated p-values. The overall model does have a low p-value though.

```
par(mfrow=c(2,2))
plot(lm.fit)
```









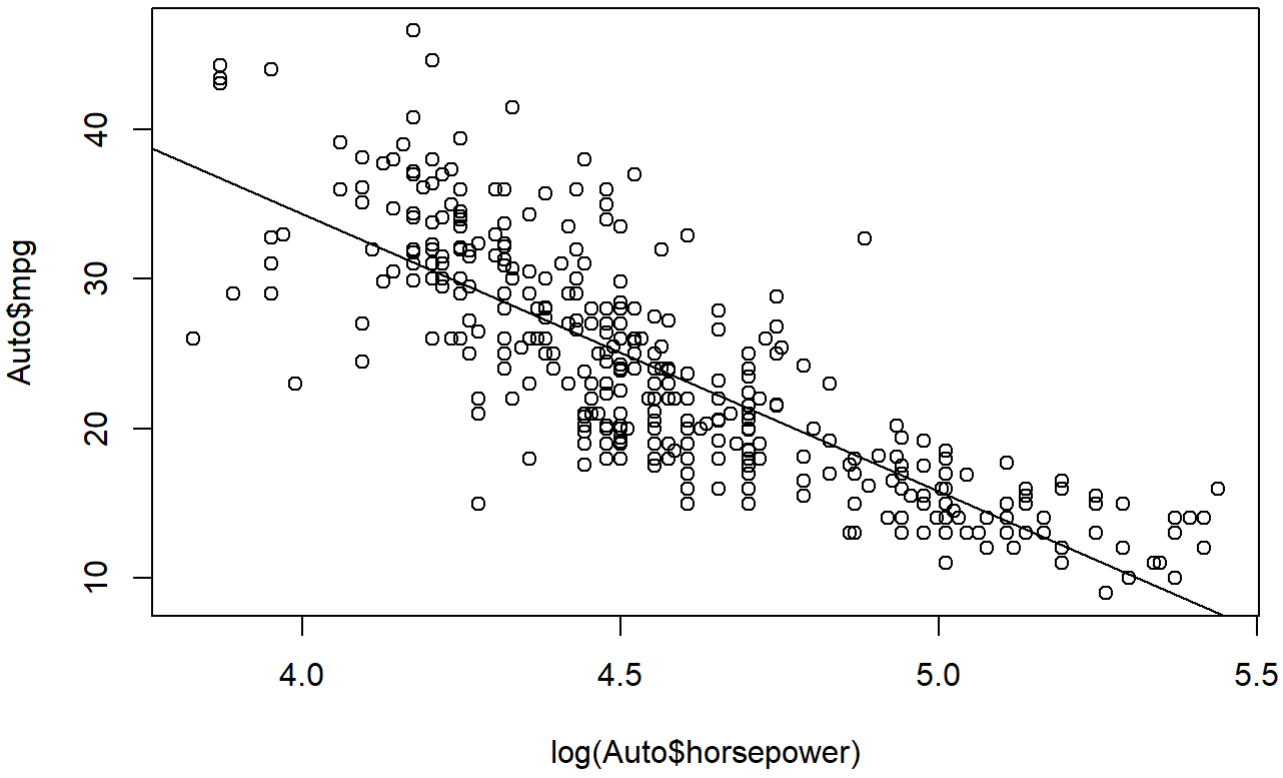
In this model, the residuals seem to be random across the fitted values, suggesting a better fit than the original regression.

f. Try a few different transformations of the variables, such as log(X), $\sqrt{X}$, X^2. Comment on your findings.

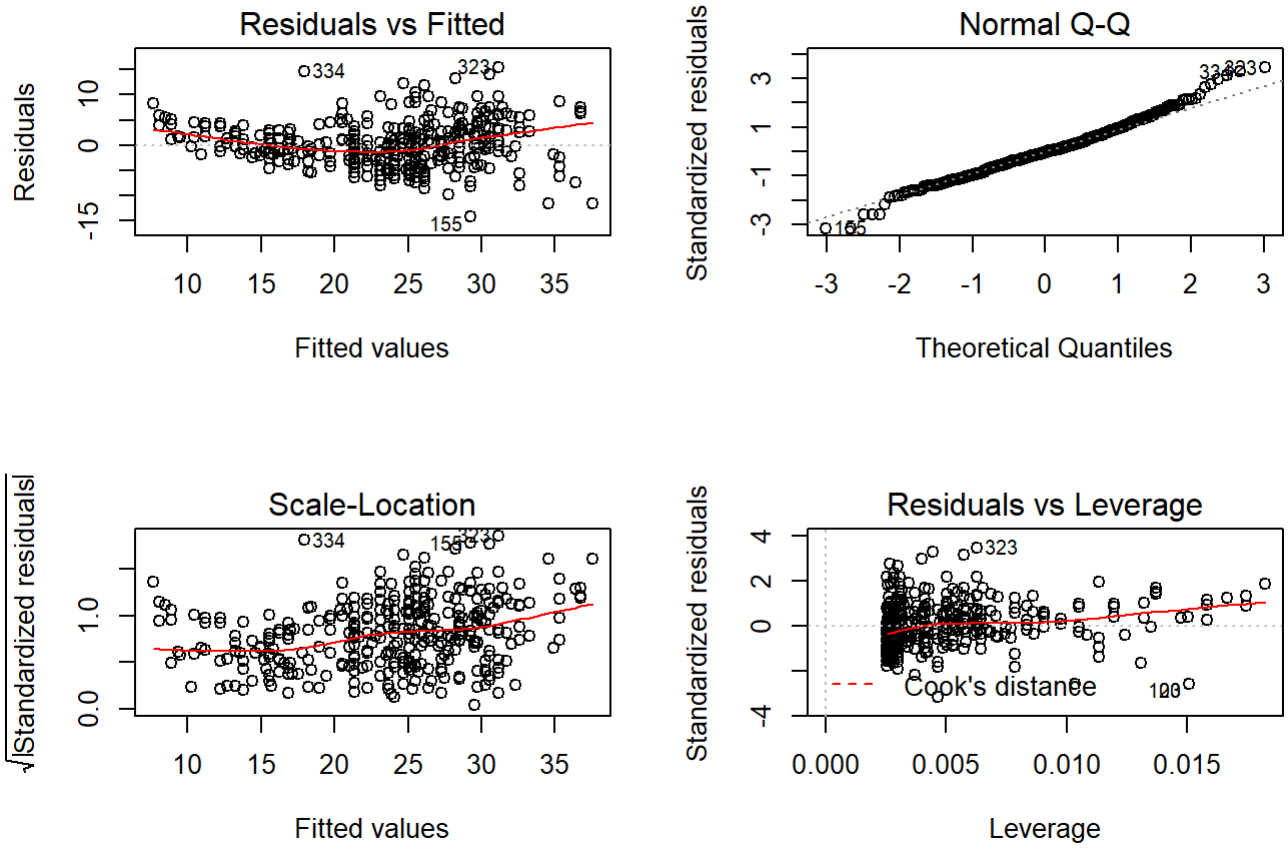
```
lm.fit = lm(mpg~log(horsepower), data=Auto)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ log(horsepower), data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2299  -2.7818  -0.2322   2.6661  15.4695
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     108.6997     3.0496   35.64   <2e-16 ***
## log(horsepower) -18.5822     0.6629  -28.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.501 on 390 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.6683, Adjusted R-squared:  0.6675
## F-statistic: 785.9 on 1 and 390 DF,  p-value: < 2.2e-16
```

```
plot(x = log(Auto$horsepower), y = Auto$mpg)
abline(lm.fit)
```



```
par(mfrow=c(2,2))
plot(lm.fit)
```



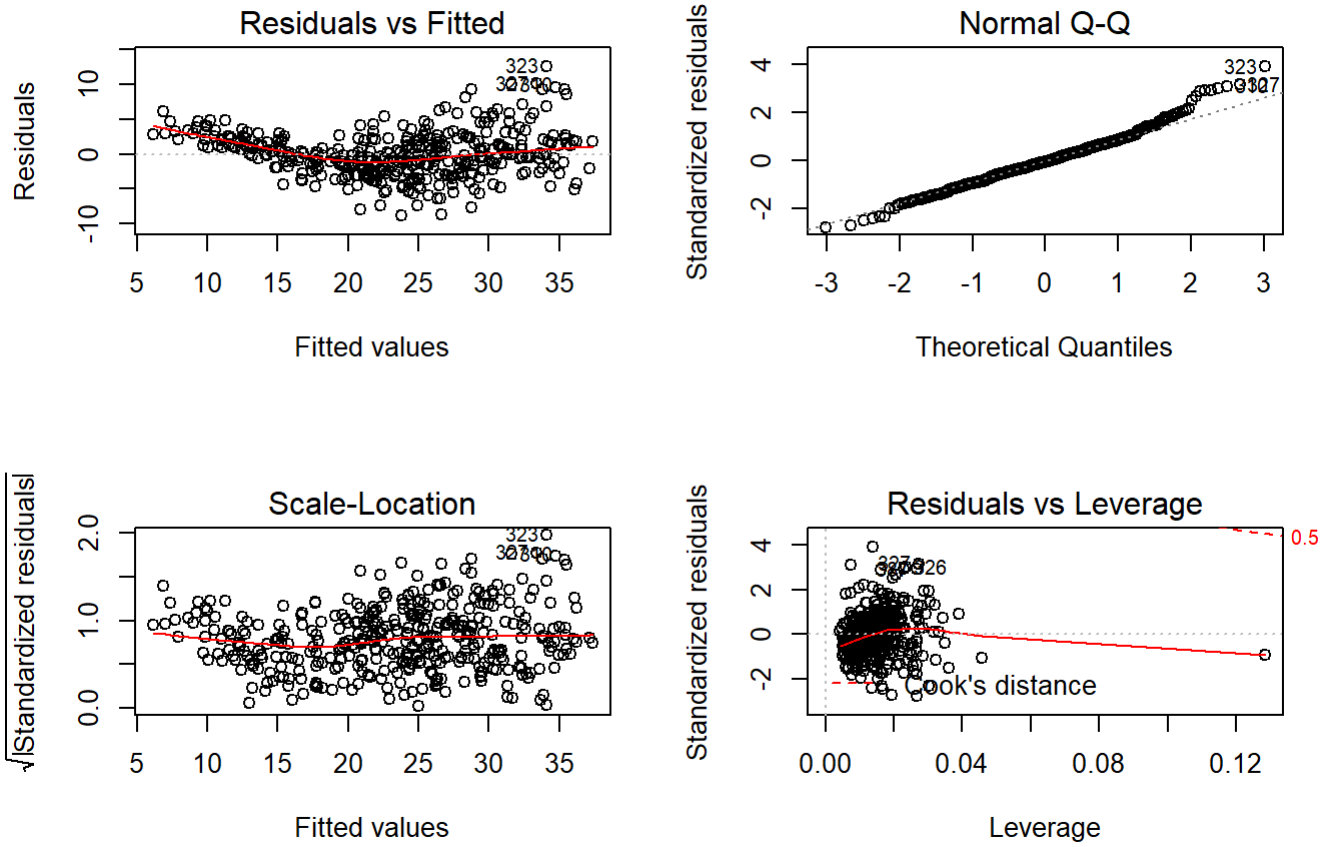
A logarithmic transform of the horsepower predictor makes a much better model for the fit from Exercise 9, but it still looks like it's not a perfect model. Let's take a look at adding that predictor to a model with the predictors that had low p-values in 9.c.ii (displacement, weight, year, and origin).

```
lm.fit = lm(mpg ~ log(horsepower) + displacement + weight + year + origin, data = Auto)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ log(horsepower) + displacement + weight +
##     year + origin, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8091 -1.9147 -0.1579  1.8765 12.5231
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      9.2041460  6.1865858   1.488  0.13763
## log(horsepower) -6.4123276  1.1128868  -5.762  1.7e-08 ***
## displacement     0.0166532  0.0049675   3.352  0.00088 ***
## weight          -0.0054967  0.0005671  -9.693  < 2e-16 ***
## year             0.7163068  0.0488094  14.676  < 2e-16 ***
## origin           1.5005416  0.2608891   5.752  1.8e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.215 on 386 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.8325, Adjusted R-squared:  0.8303
## F-statistic: 383.7 on 5 and 386 DF,  p-value: < 2.2e-16
```

In this model, all of the coefficients have low p-values, as does the overall model.

```
par(mfrow=c(2,2))
plot(lm.fit)
```



Even though all of the p-values are low, there is still an apparent pattern in the residuals plot. This pattern suggests that this model contains some bias error.

Exercise 10

This questions should be answered using the **Carseats** data set.

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.4
```

```
##
## Attaching package: 'ISLR'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     Auto
```

```
?Carseats
```

```
## starting httpd help server ...
```

```
##  done
```

a. Fit a multiple regression model to predict **Sales** using **Price**, **Urban**, and **US**.

```
lm.fit = lm(Sales ~ Price + Urban + US, data = Carseats)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Sales ~ Price + Urban + US, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9206 -1.6220 -0.0564  1.5786  7.0581
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.043469   0.651012  20.036  < 2e-16 ***
## Price       -0.054459   0.005242 -10.389  < 2e-16 ***
## UrbanYes    -0.021916   0.271650  -0.081    0.936
## USYes        1.200573   0.259042   4.635 4.86e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.472 on 396 degrees of freedom
## Multiple R-squared:  0.2393, Adjusted R-squared:  0.2335
## F-statistic: 41.52 on 3 and 396 DF,  p-value: < 2.2e-16
```

b. Provide an interpretation of each coefficient in the model. Be careful – some of the variables in the model are qualitative!

```
contrasts(Carseats$Urban)
```

```
##     Yes
## No    0
## Yes   1
```

```
contrasts(Carseats$US)
```

```
##     Yes
## No    0
## Yes   1
```

The (Intercept) coefficient of 13.043 means that if the carseats are free and the store is in a rural area outside of the US, then the predicted unit sales for the location is 13,043 carseats. Of course, that predictor value is unrealistic because the store would not just give away carseats.

The Price coefficient of -0.0544 means that for every $1 increase in the sales price, the store can expect to lose about 544 sales.

The UrbanYes coefficient of -0.219 means that, on average, urban stores sell about 219 less carseats than rural stores.

The USYes coefficient of 1.201 means that, on average, US stores sell about 1,201 more carseats than stores outside the US.

c. Write out the model in equation form, being careful to handle the qualitative variables properly.

$$Sales = \begin{cases} 14.222126 - 0.054459Price & \text{Urban store in the US} \\ 14.244042 - 0.054459Price & \text{Rural store in the US} \\ 13.021553 - 0.054459Price & \text{Urban store outside the US} \\ 13.043469 - 0.054459Price & \text{Rural store outside the US} \end{cases}$$

d. For which of the predictors can you reject the null hypothesis $H_0 : \beta_j = 0$?

For Price and US, we may reject the null hypothesis.

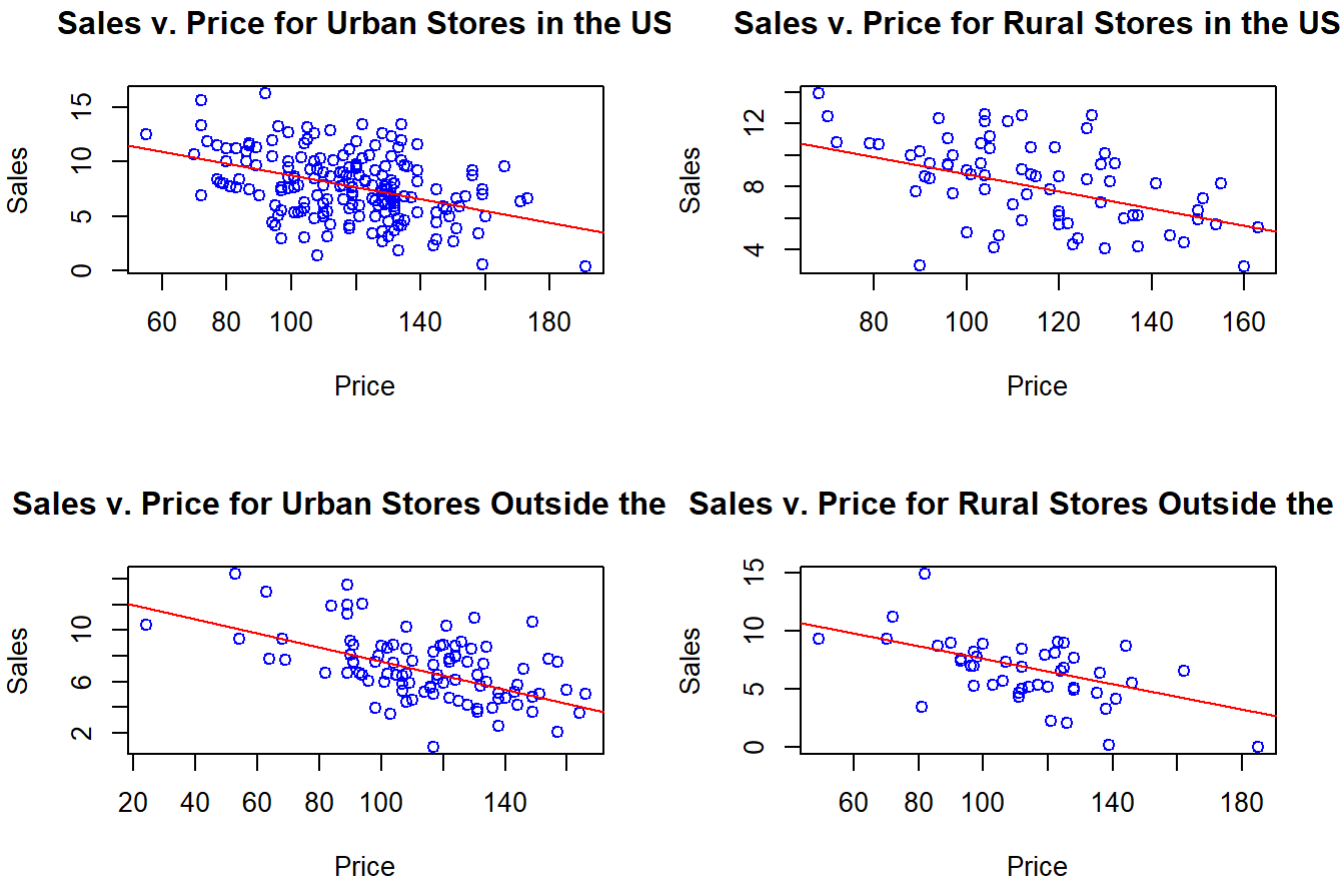
e. On the basis of your response to the previous question, fit a smaller model that only uses the predictors for which there is evidence of association with the outcome.

```
lm.fit2 = lm(Sales ~ Price + US, data = Carseats)
summary(lm.fit2)
```

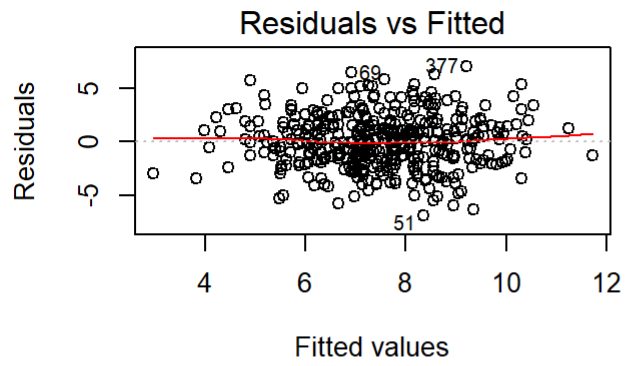
```
##
## Call:
## lm(formula = Sales ~ Price + US, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9269 -1.6286 -0.0574  1.5766  7.0515
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.03079    0.63098  20.652  < 2e-16 ***
## Price       -0.05448    0.00523 -10.416  < 2e-16 ***
## USYes        1.19964    0.25846   4.641 4.71e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.469 on 397 degrees of freedom
## Multiple R-squared:  0.2393, Adjusted R-squared:  0.2354
## F-statistic: 62.43 on 2 and 397 DF,  p-value: < 2.2e-16
```

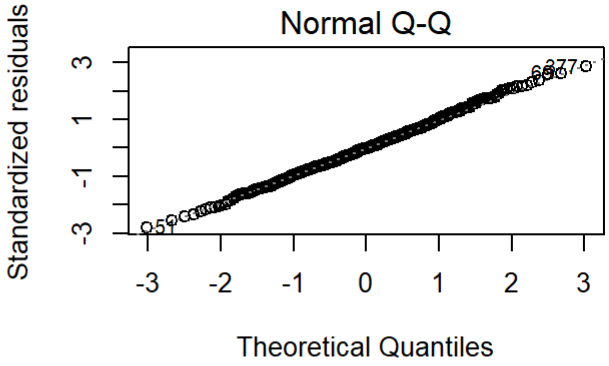
f. How well do the models in (a) and (e) fit the data?

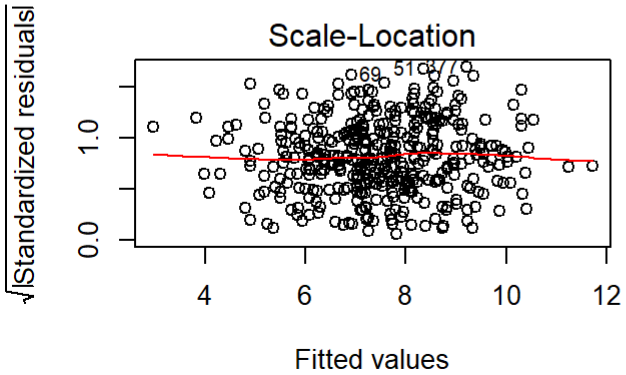
```
inUs = Carseats[Carseats$US == 'Yes',]
outUs = Carseats[Carseats$US == 'No',]
ruralInUS = inUs[inUs$Urban == 'No',]
ruralOutUS = outUs[outUs$Urban == 'No',]
urbanInUS = inUs[inUs$Urban == 'Yes',]
urbanOutUS = outUs[outUs$Urban == 'Yes',]

# Plot the first fit
par(mfrow=c(2,2))
plot(urbanInUS$Price, urbanInUS$Sales, xlab = 'Price', ylab = 'Sales', main = 'Sales v.
 Price for Urban Stores in the US', col = 'blue')
abline(14.222126, -0.054459, col = 'red')
plot(ruralInUS$Price, ruralInUS$Sales, xlab = 'Price', ylab = 'Sales', main = 'Sales v.
 Price for Rural Stores in the US', col = 'blue')
abline(14.244042, -0.054459, col = 'red')
plot(urbanOutUS$Price, urbanOutUS$Sales, xlab = 'Price', ylab = 'Sales', main = 'Sales
 v. Price for Urban Stores Outside the US', col = 'blue')
abline(13.021553, -0.054459, col = 'red')
plot(ruralOutUS$Price, ruralOutUS$Sales, xlab = 'Price', ylab = 'Sales', main = 'Sales
 v. Price for Rural Stores Outside the US', col = 'blue')
abline(13.043469, -0.054459, col = 'red')
```

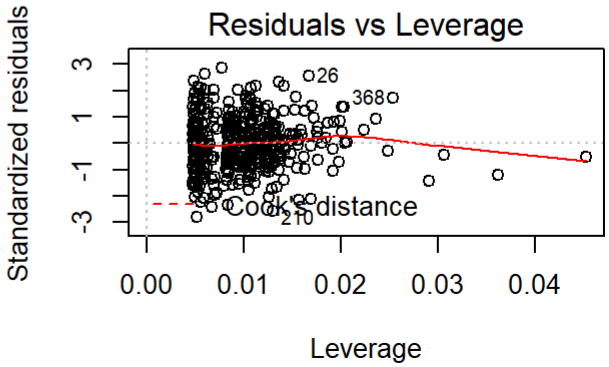



```
par(mfrow=c(2,2))
plot(lm.fit)
```



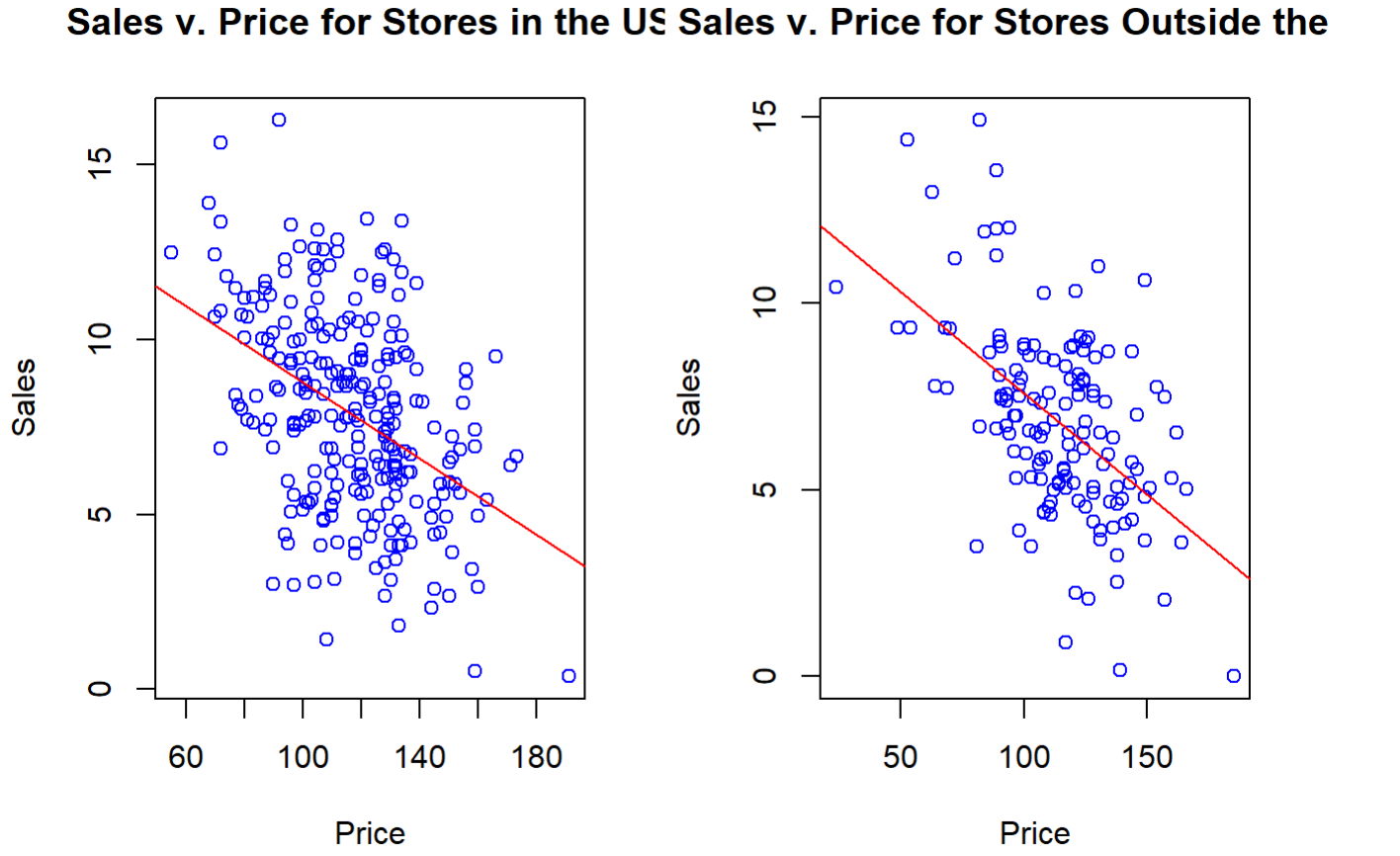




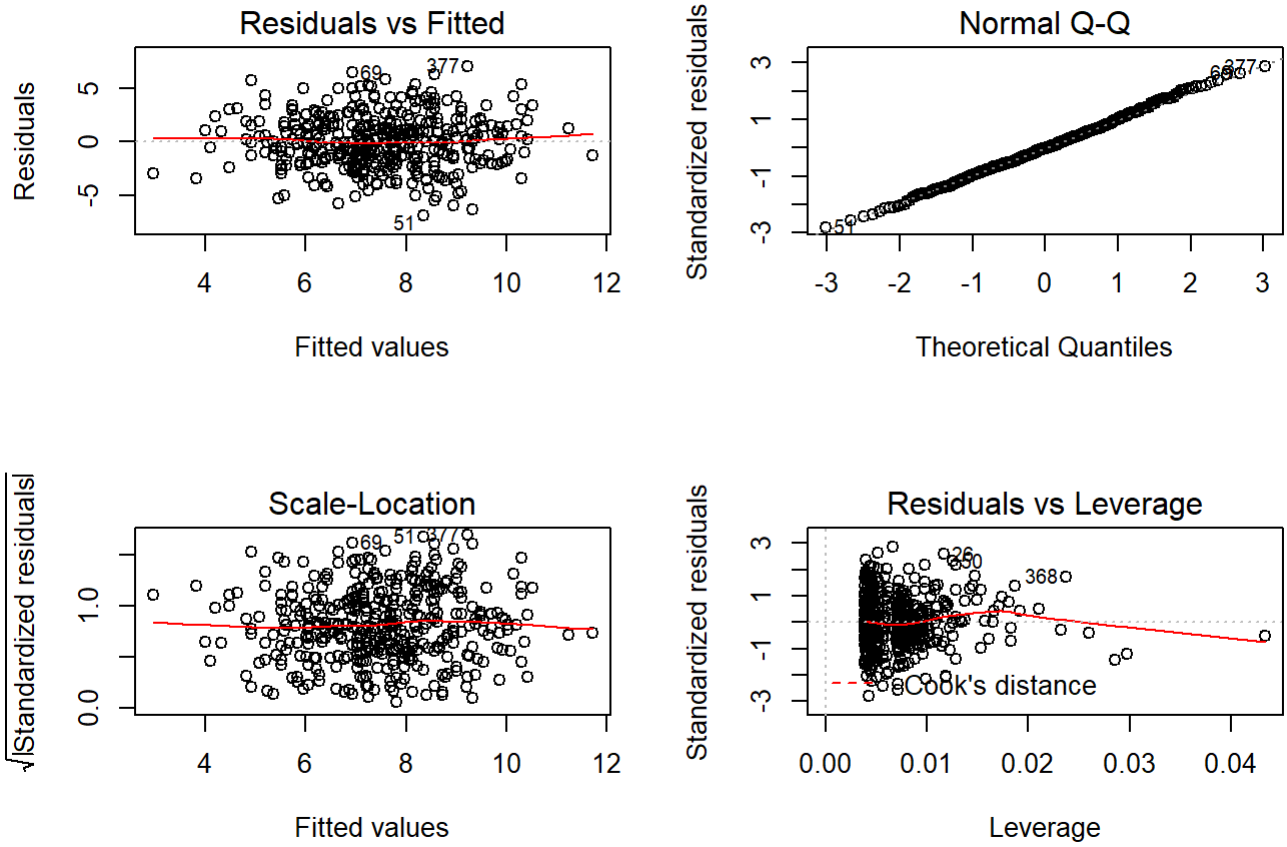


For the first model using Price, Urban, and US as predictors for Sales, there may not be a trend in the residuals, but as you can see from the plots above, as well as the R^2 value, the linear model here does not fit the data very well.

```
# Plot the first fit
par(mfrow=c(1,2))
plot(inUs$Price, inUs$Sales, xlab = 'Price', ylab = 'Sales', main = 'Sales v. Price for
 Stores in the US', col = 'blue')
abline(13.03079 + 1.19964, -0.05448, col = 'red')
plot(outUs$Price, outUs$Sales, xlab = 'Price', ylab = 'Sales', main = 'Sales v. Price fo
r Stores Outside the US', col = 'blue')
abline(13.03079, -0.05448, col = 'red')
```




```
par(mfrow=c(2,2))
plot(lm.fit2)
```



For the second model using Price and US as predictors for Sales, the residuals again seem to have no pattern. However, there is still a low R^2 value and the regression looks like it only a slightly better fit for the data.

g. Using the model from (e), obtain 95% confidence intervals for the coefficient(s).

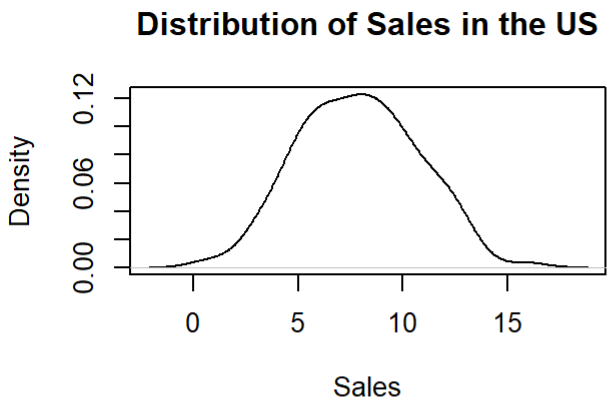
```
confint(lm.fit2)
```

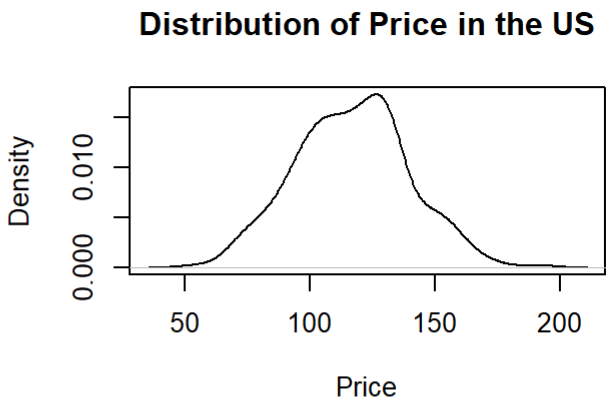
```
##                   2.5 %      97.5 %
## (Intercept) 11.79032020 14.27126531
## Price       -0.06475984 -0.04419543
## USYes        0.69151957  1.70776632
```

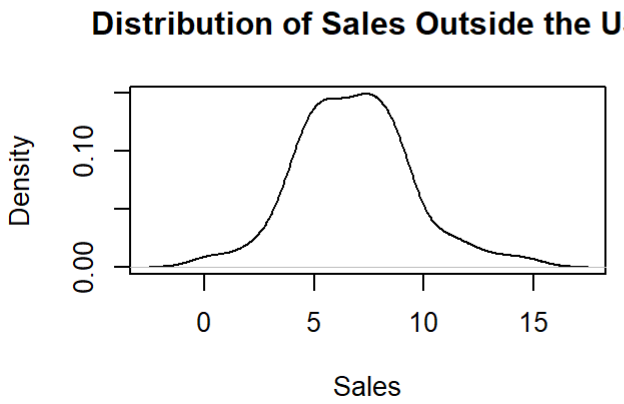
The confidence intervals are fairly wide for the estimated coefficients.

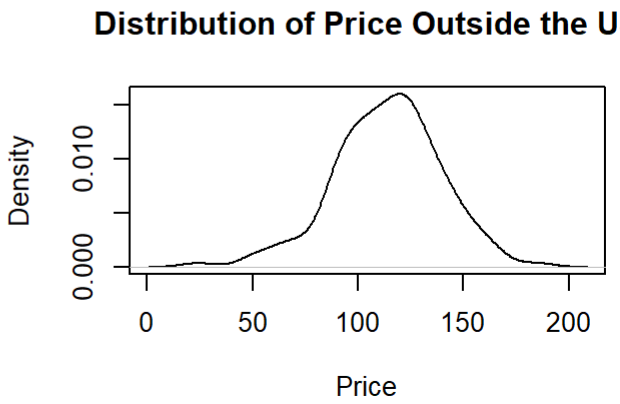
h. Is there evidence of outliers or high leverage observations in the model from (e)?

```
par(mfrow=c(2,2))
plot(density(inUs$Sales), xlab = 'Sales', ylab = 'Density', main = 'Distribution of Sale
s in the US')
plot(density(inUs$Price), xlab = 'Price', ylab = 'Density', main = 'Distribution of Pric
e in the US')
plot(density(outUs$Sales), xlab = 'Sales', ylab = 'Density', main = 'Distribution of Sal
es Outside the US')
plot(density(outUs$Price), xlab = 'Price', ylab = 'Density', main = 'Distribution of Pri
ce Outside the US')
```









You can see that the Price distribution in the US is skewed right. The observations with Price > $175 for stores in the US may be considered high leverage points. The Price distribution outside the US is skewed left. The observations with Price < $50 for stores outside the US may be considered high leverage points. There don't seem to be any outliers for stores within the US, but the distribution of Sales for stores outside the US has long tails. I would consider any of the observations with Sales > 15 for stores outside the US to be outliers. I don't know what the meaning behind negative sales is, but it seems like any observation with Sales < 0 for stores outside the US could be considered outliers.

```
# High leverage points
Carseats[(Carseats$Price > 175 & Carseats$US == 'Yes') | (Carseats$Price < 50) & Carseat
s$US == 'No', c('Sales', 'Price', 'US')]
```

```
##     Sales Price  US
## 43  10.43    24  No
## 126  9.34    49  No
## 166  0.37   191 Yes
```

So it looks like we have 3 points that could be considered high leverage.

```
# Outliers
Carseats[Carseats$US == 'No' & (Carseats$Sales > 15 | Carseats$Sales < 0), c('Sales', 'P
rice', 'US')]
```

```
## [1] Sales Price US
## <0 rows> (or 0-length row.names)
```

I guess the plot is a little misleading, because there aren't actually any stores outside the US that have more than 15 or less than 0 sales. Therefore, I would say that there aren't really any outliers in the data.

Another approach to determining outliers and high leverage points, which is more applicable to models with more predictors, is to use the hatvalues *and* rstandard *functions in R.*

```
p = 2
n = length(Carseats)
leverage = hatvalues(lm.fit2)
print(sum(leverage > (p + 1)/n))
```

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

According to the definition of a high leverage point from section 3.3.3, there are no high leverage points in the data.

```
studentizedResiduals = rstudent(lm.fit2)
print(sum(studentizedResiduals >= 3))
```

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

According to the definition of an outlier from section 3.3.3, there are no outliers in the data.

Exercise 11

In this problem we will investigate the t-statistic for the null hypothesis: $H_0 : \beta = 0$ in simple linear regression without an intercept. To begin, we generate a predictor **x** and a response **y** as follows.

```
> set.seed(1)
> x=rnorm(100)
> y=2*x+rnorm(100)
```

```
set.seed(1)
x = rnorm(100)
y = 2 * x + rnorm(100)
```

a. Perform a simple linear regression of **y** onto **x**, *without* an intercept. Report the coefficient estimate $\hat{\beta}$, the standard error this coefficient estimate, its standard error, and the corresponding t-statistic and p-values associated with the null hypothesis $H_0 : \beta = 0$. Comment on these results. (You can perform regression without an intercept using the command **lm(y~x+0)**.)

```
lm.fit = lm(y ~ x + 0)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x + 0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9154 -0.6472 -0.1771  0.5056  2.3109
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x   1.9939     0.1065   18.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9586 on 99 degrees of freedom
## Multiple R-squared:  0.7798, Adjusted R-squared:  0.7776
## F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

c. What is the relationship between the results obtained in (a) and (b)?

The true relationship is y = 2x. The linear fit is very close at y = 1.9939x.

d. For the regression of Y onto X without an intercept, the t-statistic for $H_0 : \beta = 0$ takes the form $\frac{\hat{\beta}}{SE(\hat{\beta})}$, where $\hat{\beta}$ is given by (3.38), and where

$$SE(\hat{\beta}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2}{(n-1)\sum_{i'=1}^{n} x_{i'}^2}}$$

. (These formulas are slightly different from those given in Sections 3.1.1 and 3.1.2, since here we are performing regression without an intercept.) Show algebraically, and confirm numerically in R, that the t-statistic can be written as

$$\frac{(\sqrt{n-1})\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i'=1}^{n} y_{i'}^2 - (\sum_{i'=1}^{n} x_{i'} y_{i'})^2}}$$

.

```
n = 100
beta = as.numeric(coef(lm.fit)['x'])
se = sqrt(sum((y - x * beta)^2) / ((n - 1) * sum(x*x)))
tstat1 = beta / se
tstat2 = sqrt(n - 1) * sum(x * y) / sqrt(sum(x*x) * sum(y * y) - (sum(x*y))^2)
tstat1 == tstat2
```

```
## [1] TRUE
```

The values are numerically equal in R.

Proof for algebraic congruency:

First, we must define $\hat{\beta}$ for a simple linear regression with no intercept.

Typically, $\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$ because we want to find what change in y results from a change in x and we must center those deltas around the means to account for β_0. However, if we assume that $\beta_0 = 0$, then we can center those deltas around 0. Therefore, $\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$ for a simple linear regression with no intercept.

Plugging that into the equation…

$$\frac{\hat{\beta}}{SE(\hat{\beta})} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} \sqrt{\frac{(n-1)\sum_{i=1}^{n} x_i^2}{\sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2}}$$

$$= \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2}}$$

$$= \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n}(y_i^2 - 2\hat{\beta} y_i x_i + \hat{\beta}^2 x_i^2)}}$$

$$= \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} x_i^2)\hat{\beta}(2y_i x_i - \hat{\beta} x_i^2)}}$$

$$= \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} x_i^2)(\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2})(2\sum_{i=1}^{n} y_i x_i - (\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}) x_i^2)}}$$

$$= \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} x_i y_i)(2\sum_{i=1}^{n} y_i x_i - \sum_{i=1}^{n} x_i y_i)}}$$

$$= \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} x_i y_i)(\sum_{i=1}^{n} x_i y_i)}}$$

$$= \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} x_i y_i)^2}}$$

e. Using the results from (d), argue that the t-statistic for the regression of **y** onto **x** is the same as the t-statistic for the regression of **x** onto **y**.

If you use y *as the predictor and* x *as the response, the equation in (e) is the same.*

$$\frac{\sqrt{n-1}\sum_{i=1}^{n} y_i x_i}{\sqrt{\sum_{i=1}^{n} y_i^2 \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} y_i x_i)^2}} = \frac{\sqrt{n-1}\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} x_i y_i)^2}}$$

Therefore, the t-statistics are the same for both regressions.

f. In R, show that when regression is performed *with* an intercept, the t-statistic for $H_0 : \beta_1 = 0$ is the same for the regression of **y** onto **x** as it is for the regression of **x** onto **y**.

```
getT <- function(x, y){
  n = length(x)
  lm.fit = lm(y ~ x)
  beta1 = as.numeric(coef(lm.fit)['x'])
  RSS = sum((y - predict(lm.fit, data.frame(x=x)))^2)
  RSE = sqrt(RSS / (n - 2))
  SE = sqrt(RSE^2 / sum((x - mean(x))^2))
  return(beta1/SE)
}
getT(x,y)
```

```
## [1] 18.5556
```

```
getT(y,x)
```

```
## [1] 18.5556
```

You can see that the t-statistic is the same for both the regression of y *onto* x *and for* x *onto* y.

Exercise 12

This problem involves simple linear regression without an intercept.

a. Recall that the coefficient estimate $\hat{\beta}$ for the linear regression of Y onto X without an intercept is given by (3.38). Under what circumstance is the coefficient estimate for the regression of X onto Y the same as the coefficient estimate for the regression of Y onto X?

For a simple linear regression with no intercept, $\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$ *Therefore, the coefficient estimate for the regression of X onto Y is the same as the coefficient estimate for the regression of Y onto X iff* $\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i^2$

b. Generate an example in R with *n* = 100 observations in which the coefficient estimate for the regression X onto Y is *different from* the coefficient estimate for the regression onto X.

```
y = 1:100 * 2
x = 1:100 - 4
```

```
summary(lm(y~x+0))
```

```
##
## Call:
## lm(formula = y ~ x + 0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9219 -0.8483  2.2253  5.2989  8.3726
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x 2.124186   0.007748   274.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.241 on 99 degrees of freedom
## Multiple R-squared:  0.9987, Adjusted R-squared:  0.9987
## F-statistic: 7.516e+04 on 1 and 99 DF,  p-value: < 2.2e-16
```

```
summary(lm(x~y+0))
```

```
##
## Call:
## lm(formula = x ~ y + 0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9403 -2.4627 -0.9851  0.4925  1.9701
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## y 0.470149   0.001715   274.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.995 on 99 degrees of freedom
## Multiple R-squared:  0.9987, Adjusted R-squared:  0.9987
## F-statistic: 7.516e+04 on 1 and 99 DF,  p-value: < 2.2e-16
```

c. Generate an example in R with *n* = 100 observations in which the coefficient estimate for the regression of X onto Y is *the same as* the coefficient estimate for the regression of Y onto X.

```
x = 1:100
y = 1:100 * -1
```

```
summary(lm(y~x+0))
```

```
## Warning in summary.lm(lm(y ~ x + 0)): essentially perfect fit: summary may
## be unreliable
```

```
##
## Call:
## lm(formula = y ~ x + 0)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.294e-14 -2.218e-15 -2.900e-17  2.094e-15  3.082e-13
##
## Coefficients:
##     Estimate Std. Error    t value Pr(>|t|)
## x -1.000e+00  5.379e-17 -1.859e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.129e-14 on 99 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.457e+32 on 1 and 99 DF,  p-value: < 2.2e-16
```

```
summary(lm(x~y+0))
```

```
## Warning in summary.lm(lm(x ~ y + 0)): essentially perfect fit: summary may
## be unreliable
```

```
##
## Call:
## lm(formula = x ~ y + 0)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.082e-13 -2.094e-15  2.900e-17  2.218e-15  1.294e-14
##
## Coefficients:
##     Estimate Std. Error    t value Pr(>|t|)
## y -1.000e+00  5.379e-17 -1.859e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.129e-14 on 99 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.457e+32 on 1 and 99 DF,  p-value: < 2.2e-16
```

Exercise 13

In this exercise you will create some simulated data and will fit simple linear regression models to it. Make sure to use **set.seed(1)** prior to starting part (a) to ensure consitent results.

```
set.seed(1)
```

a. Using the **rnorm()** function, create a vector, **x**, containing 100 observations drawn from a *N(0,1)* distribution. This represents a feature, X.

```
x = rnorm(100)
```

b. Using the **rnorm()** function, create a vector, **eps**, containing 100 observations drawn from a *N(0,0.25)* distribution (i.e. a normal distribution with mean zero and variance 0.25).

```
eps = rnorm(100, sd = .25)
```

c. Using **x** and **eps**, generate a vector **y** according to the model $Y = -1 + 0.5X + \epsilon$. What is the length of the vector **y**? What are the values of β_0 and β_1 in this linear model?

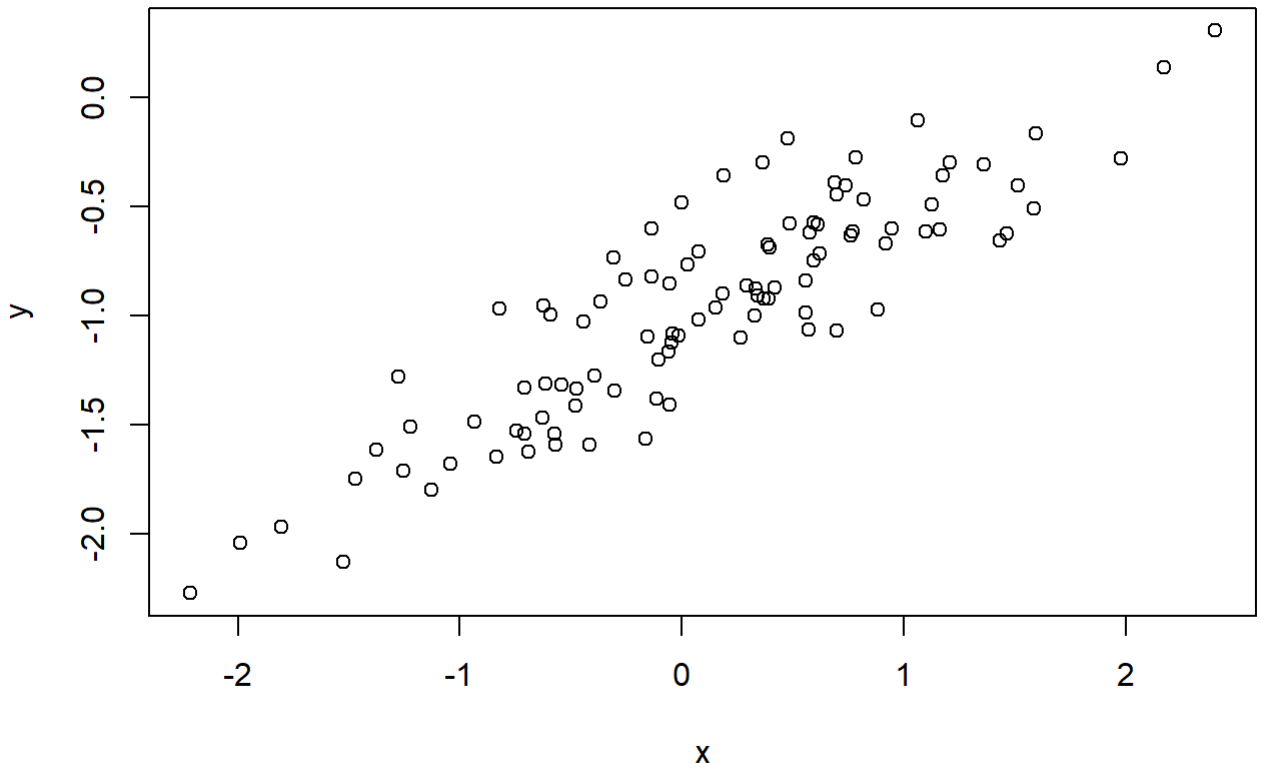
```
y = -1 + .5 * x + eps
length(y)
```

```
## [1] 100
```

y is a vector with 100 observations. The true parameters of the model are: $\beta_0 = -1$ *and* $\beta_1 = 0.5$.

d. Create a scatterplot displaying the relationship between **x** and **y**. COmment on what you observe.

```
plot(x,y)
```



There is clearly a relationship between x *and* y *with a little bit of noise.*

e. Fit a least squares linear model to predict **y** using **x**. Comment on the model obtained. How do the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ compare to the true values of β_0 and β_1?

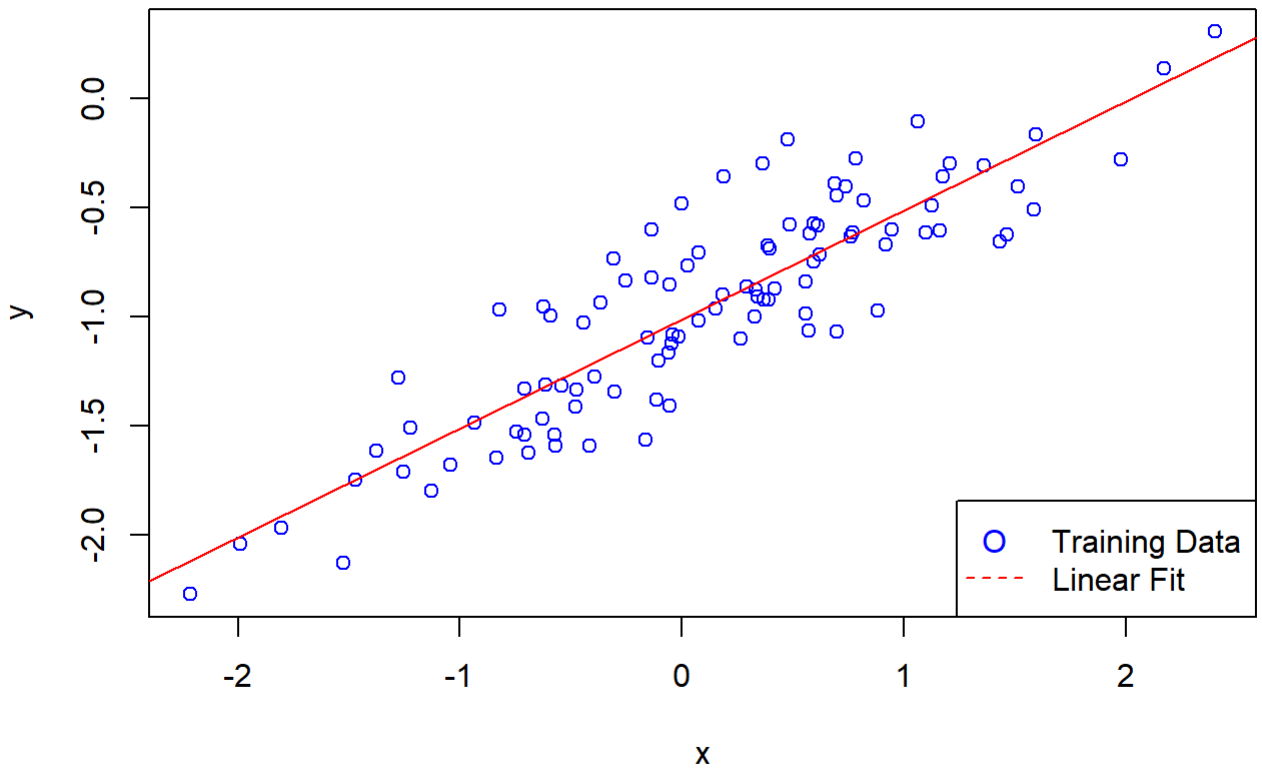
```
lm.fit = lm(y ~ x)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46921 -0.15344 -0.03487  0.13485  0.58654
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.00942    0.02425  -41.63   <2e-16 ***
## x            0.49973    0.02693   18.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2407 on 98 degrees of freedom
## Multiple R-squared:  0.7784, Adjusted R-squared:  0.7762
## F-statistic: 344.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

The estimated parameters for the model are: $\hat{\beta}_0 = -1.00942$ *and* $\hat{\beta}_1 = 0.49973$. *These values are very close to the true relationship.*

f. Display the least squares line on the scatterplot obtained in (d). Draw the population regression line on the plot, in a different color. Use the **legend()** command to create an appropriate legend.

```
plot(x,y, col = 'blue')
abline(lm.fit, col = 'red')
legend('bottomright', legend = c('Training Data', 'Linear Fit'), lty = c(NA,2), pch = c(
'O', NA), col = c('blue', 'red'))
```



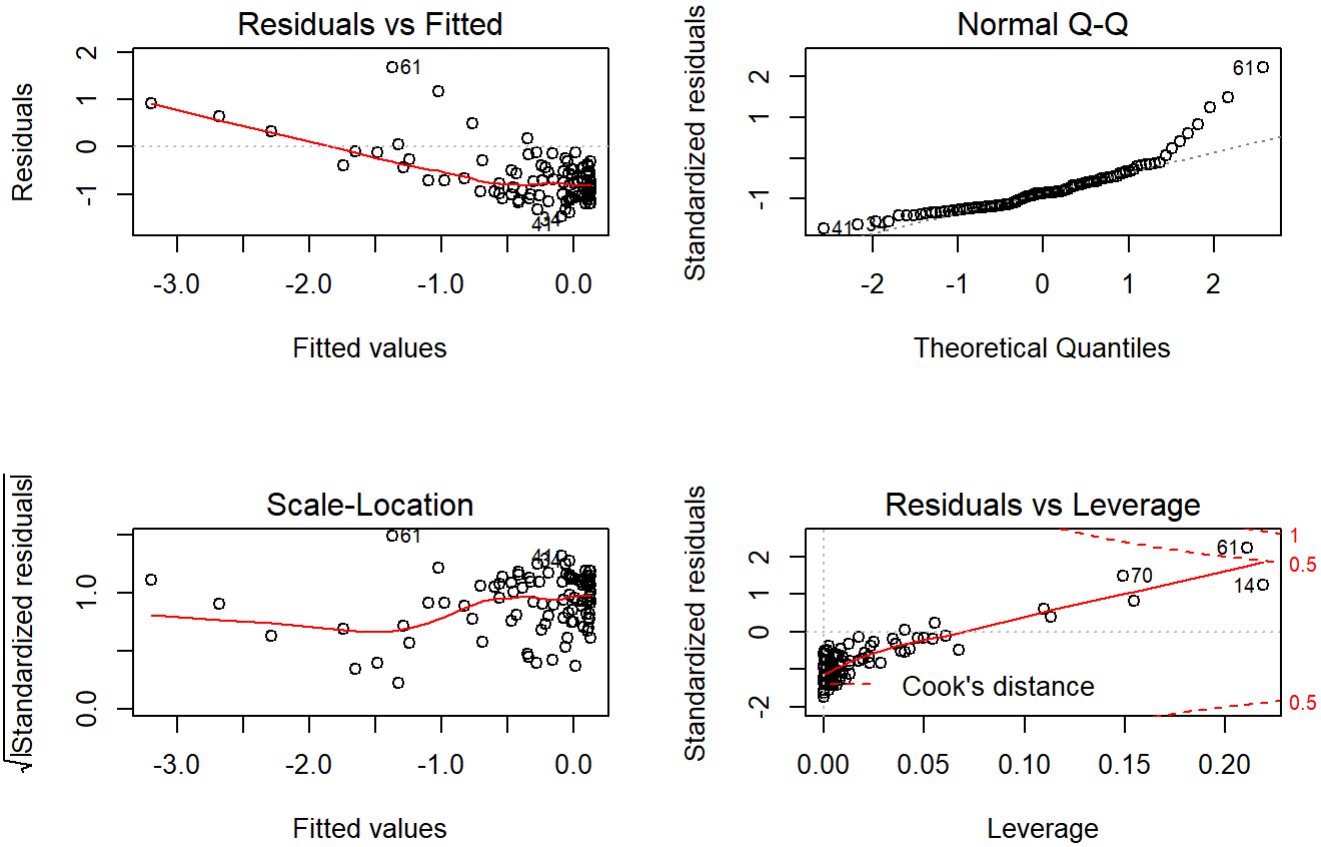
g. Now fit a polynomial regression model that predicts **y** using **x** and $\mathbf{x^2}$. Is there evidence that the quadratic term improves the model fit? Explain your answer.

```
lm.fit2 = lm(y~I(x^2)+x+0)
```

```
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = y ~ I(x^2) + x + 0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4705 -1.0175 -0.7383 -0.4404  1.6750
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## I(x^2) -0.43595    0.06134  -7.107 1.93e-10 ***
## x       0.47742    0.09522   5.014 2.37e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8452 on 98 degrees of freedom
## Multiple R-squared:  0.4008, Adjusted R-squared:  0.3886
## F-statistic: 32.78 on 2 and 98 DF,  p-value: 1.26e-11
```

```
par(mfrow=c(2,2))
plot(lm.fit2)
```



The p-value for the x^2 term is very low, which does suggest that a quadratic model would improve the fit. However, the residuals definitely have a downward trend, which tells us that there is probably some bias error in the model. Of course since we know that the true relationship is linear, it is obvious that there would be bias error associated

with a quadratic fit.

h. Repeat (a)-(f) after modifying the data generation process in such a way that there is *less* noise in the data. The model from (c) should remain the same. You can do this by decreasing the variance of the normal distribution used to generate the error term ϵ in (b). Describe your results.

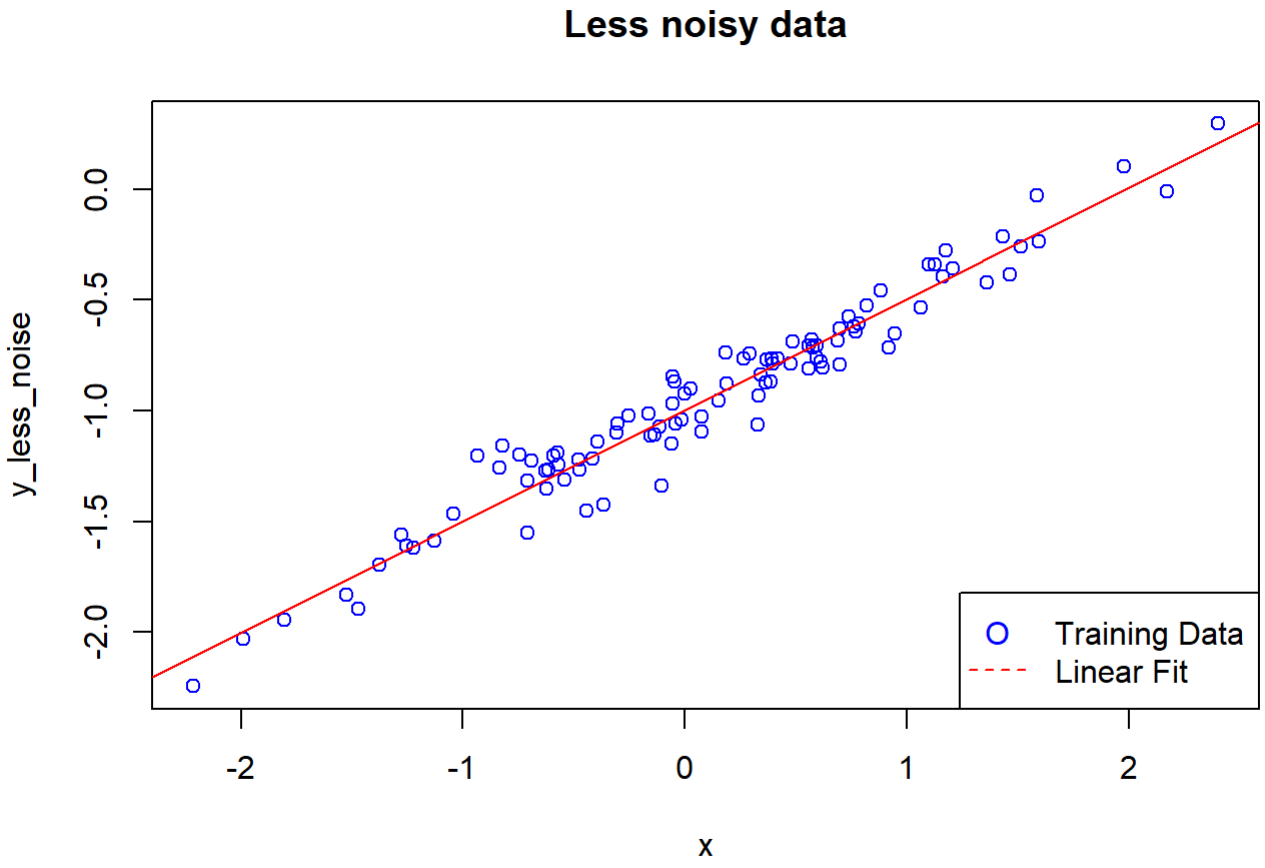
```
y_less_noise = -1 + .5 * x + rnorm(100, sd = .1)
lm.fit_less_noise = lm(y_less_noise ~ x)
```

```
summary(lm.fit_less_noise)
```

```
##
## Call:
## lm(formula = y_less_noise ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.291411 -0.048230 -0.004533  0.064924  0.264157
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.99726    0.01047  -95.25   <2e-16 ***
## x            0.50212    0.01163   43.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1039 on 98 degrees of freedom
## Multiple R-squared:  0.9501, Adjusted R-squared:  0.9495
## F-statistic:  1864 on 1 and 98 DF,  p-value: < 2.2e-16
```

Both the estimated parameters are closer to their true values.

```
plot(x,y_less_noise, col = 'blue', main = 'Less noisy data')
abline(lm.fit_less_noise, col = 'red')
legend('bottomright', legend = c('Training Data', 'Linear Fit'), lty = c(NA,2), pch = c(
'O', NA), col = c('blue', 'red'))
```



You can see less variance in the training data and a tighter fit to the estimated model.

i. Repeat (a)-(f) after modifying the generation process in such a way that there is *more* noise in the data. Describe your results.

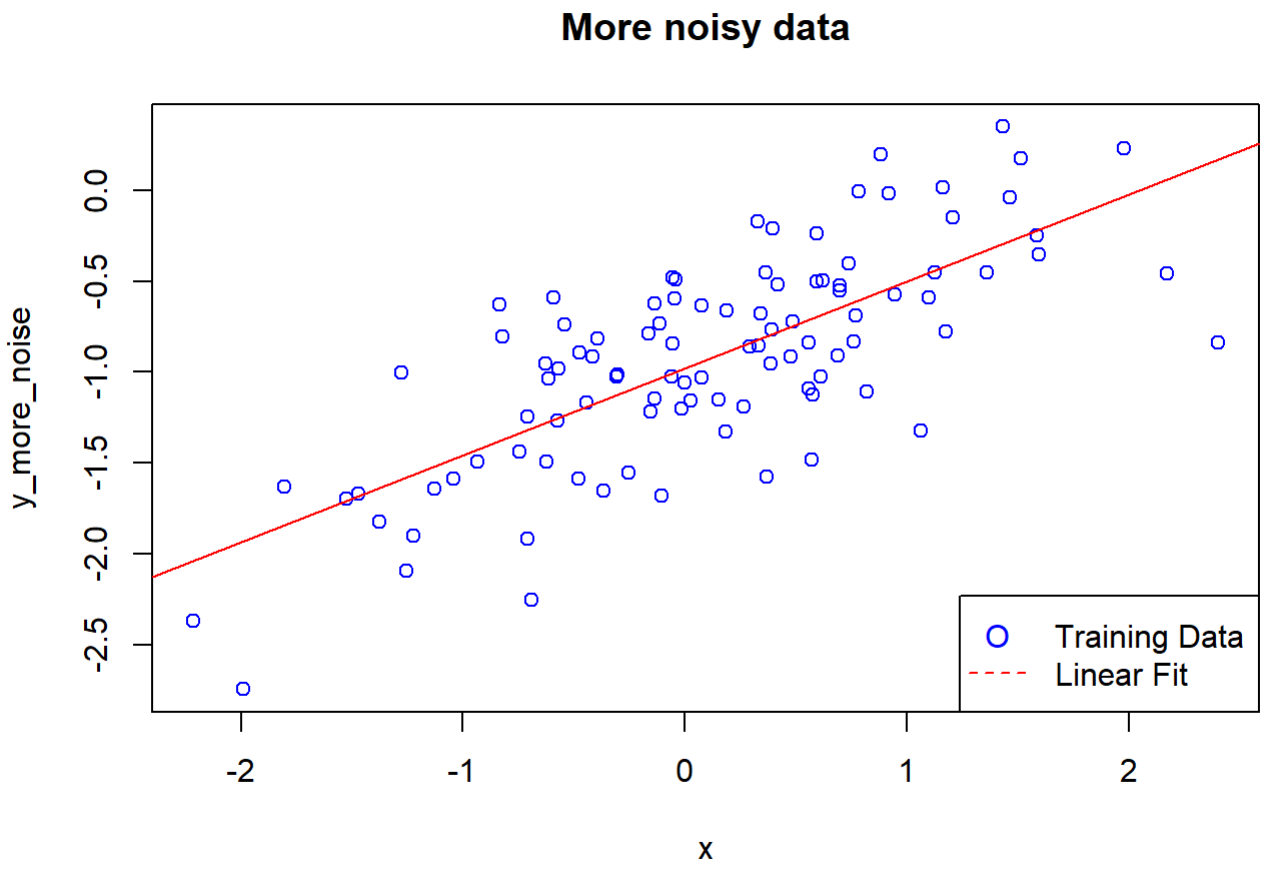
```
y_more_noise = -1 + .5 * x + rnorm(100, sd = .4)
lm.fit_more_noise = lm(y_more_noise ~ x)
```

```
summary(lm.fit_more_noise)
```

```
##
## Call:
## lm(formula = y_more_noise ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0065 -0.2181 -0.0151  0.2692  0.7516
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.97693    0.04011  -24.36   <2e-16 ***
## x            0.47773    0.04455   10.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3982 on 98 degrees of freedom
## Multiple R-squared:  0.5398, Adjusted R-squared:  0.5351
## F-statistic:   115 on 1 and 98 DF,  p-value: < 2.2e-16
```

Both the estimated parameters are farther from their true values.

```
plot(x,y_more_noise, col = 'blue', main = 'More noisy data')
abline(lm.fit_more_noise, col = 'red')
legend('bottomright', legend = c('Training Data', 'Linear Fit'), lty = c(NA,2), pch = c(
'O', NA), col = c('blue', 'red'))
```



You can see more variance in the training data and a looser fit to the estimated model.

j. What are the confidence intervals for β_0 and β_1 based on the original data set, the noisier data set, and the less noisy data set? Comment on your results.

```
print('Original')
```

```
## [1] "Original"
```

```
print(confint(lm.fit))
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.0575402 -0.9613061
## x            0.4462897  0.5531801
```

```
print('Noisier')
```

```
## [1] "Noisier"
```

```
print(confint(lm.fit_more_noise))
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.0565360 -0.8973317
## x            0.3893088  0.5661423
```

```
print('Less noisy')
```

```
## [1] "Less noisy"
```

```
print(confint(lm.fit_less_noise))
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.0180413 -0.9764850
## x            0.4790377  0.5251957
```

Unsurprisingly, the confidence intervals get wider as more noise is introduced to the training data. The increase in variance seems to have a greater impact on the model than the decrease in variance does.

Exercise 14

a. Perform the following commands in R:

```
> set.seed(1)
> x1=runif(100)
> x2=0.5*x1+rnorm(100)/10
> y=2+2*x1+0.3*x2+rnorm(100)
```

```
set.seed(1)
x1 = runif(100)
x2 = .5 * x1 + rnorm(100) / 10
y = 2 + 2 * x1 + .3 * x2 + rnorm(100)
```

The last line corresponds to creating a linear model in which **y** is a function of **x1** and **x2**. Write out the form of the linear model. What are the regression coefficients?

The true parameters of the linear relationship are $\beta_0 = 2$, $\beta_1 = 2$, and $\beta_2 = 0.3$.

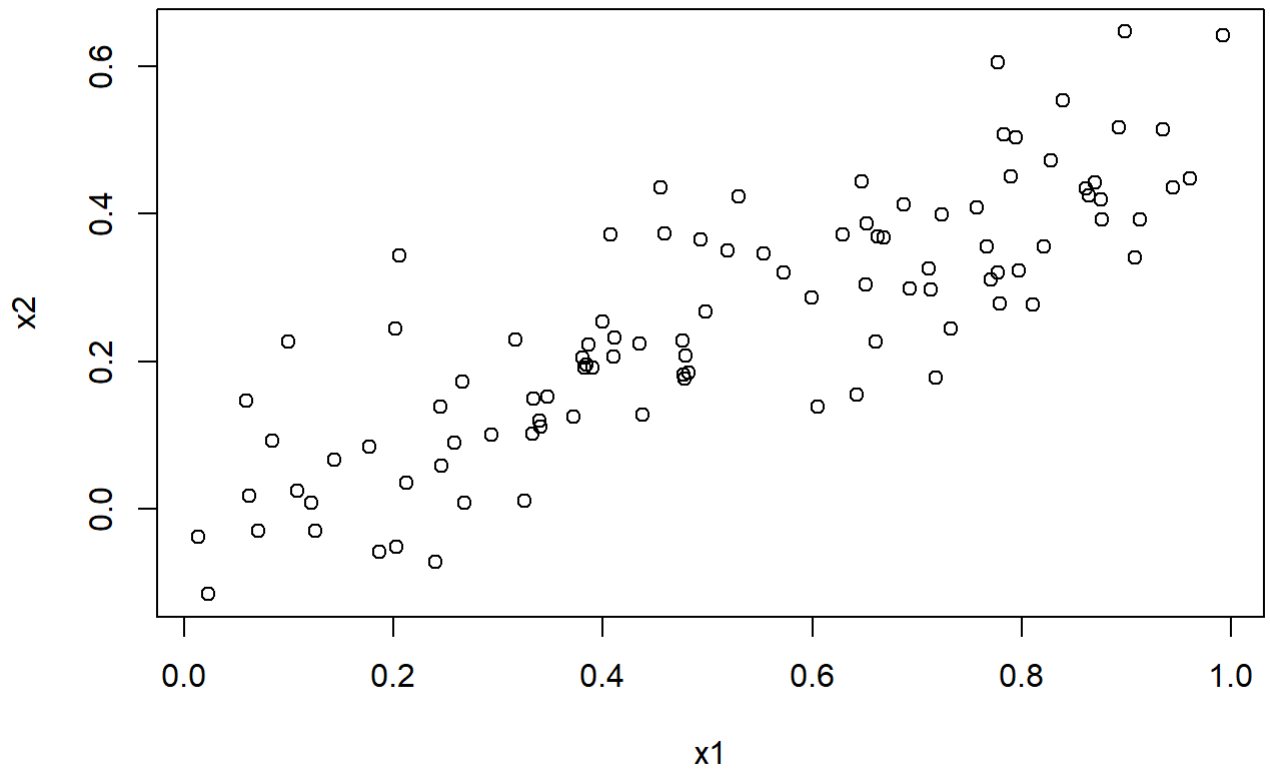
b. What is the correlation between **x1** and **x2**? Create a scatterplot displaying the relationship between the variables.

```
cor(x1,x2)
```

```
## [1] 0.8351212
```

The correlation between x1 *and* x2 *is 0.835.*

```
plot(x1, x2)
```



There is a high correlation between x1 *and* x2. *There is also clear evidence of a relationship in the scatterplot. These both make sense since we used* x1 *to define* x2 *via a linear relationship.*

c. Using this data, fit a least squares regression to predict **y** using **x1** and **x2**. Describe the results obtained. What are $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$? How do these relate to the true β_0, β_1, and β_2? Can you reject the null hypothesis $H_0 : \beta_1 = 0$? How about the null hypothesis $H_0 : \beta_2 = 0$?

```
lm.fit = lm(y~x1+x2)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8311 -0.7273 -0.0537  0.6338  2.3359
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1305     0.2319   9.188 7.61e-15 ***
## x1            1.4396     0.7212   1.996   0.0487 *
## x2            1.0097     1.1337   0.891   0.3754
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.056 on 97 degrees of freedom
## Multiple R-squared:  0.2088, Adjusted R-squared:  0.1925
## F-statistic:  12.8 on 2 and 97 DF,  p-value: 1.164e-05
```

The estimated parameters for the linear model are $\hat{\beta}_0 = 2.1305$, $\hat{\beta}_1 = 1.4396$, and $\hat{\beta}_2 = 01.0097$. *These estimated values are not very close to the true parameters of the relationship. Furthermore, the p-value for* $\hat{\beta}_2$ *is not sufficiently low to reject the null hypothesis. However, we may reject the null hypothesis for* $\hat{\beta}_0$ *and* $\hat{\beta}_1$ *(*$\hat{\beta}_1$ *may only be rejected using 5% as the threshold, but not 1% or less). The discrepancies between the true relationship and the model here likely result from the fact that* x2 *is directly related to* x1.

d. Now fit a least squares regression to predict **y** using only **x1**. Comment on your results. Can you reject the null hypothesis $H_0 : \beta_1 = 0$?

```
lm.fit1 = lm(y~x1)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = y ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89495 -0.66874 -0.07785  0.59221  2.45560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1124     0.2307   9.155 8.27e-15 ***
## x1            1.9759     0.3963   4.986 2.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.055 on 98 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.1942
## F-statistic: 24.86 on 1 and 98 DF,  p-value: 2.661e-06
```

The p-value is low enough to reject the null hypothesis. In fact, it is even lower than it was in the model that incorporated x2. *Also, the estimated value for* $\hat{\beta}_1$ *is closer to the true value.*

e. Now fit a least squares regression to predict **y** using only **x2**. Comment on your results. Can you reject the null hypothesis $H_0 : \beta_2 = 0$?

```
lm.fit2 = lm(y~x2)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = y ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## x2            2.8996     0.6330    4.58 1.37e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

The p-value here is low enough to reject the null hypothesis. In this case it is much lower than it was in the original model. However, both the estimated values for $\hat{\beta}_0$ *and* $\hat{\beta}_2$ *are even further away from their true values than they were in the original model.*

f. Do the results obtained in (c) - (e) contradict each other? Explain your answer.

The large value for $\hat{\beta}_2$ *in (e) actually makes sense, because that coefficient is compensating for the lack of* x1 *in the model. Since* x2 *is defined directly by* x1, *increasing the* $\hat{\beta}_2$ *parameter to account for the contribution of the* x1 *predictor to the response value can actually produce a fairly accurate model for predicting response values. However, it can also lead to incomplete and possibly incorrect inferences about the predictor-response relationship. Therefore, the viability of the model produced in (e) depends on the purpose of the model.*

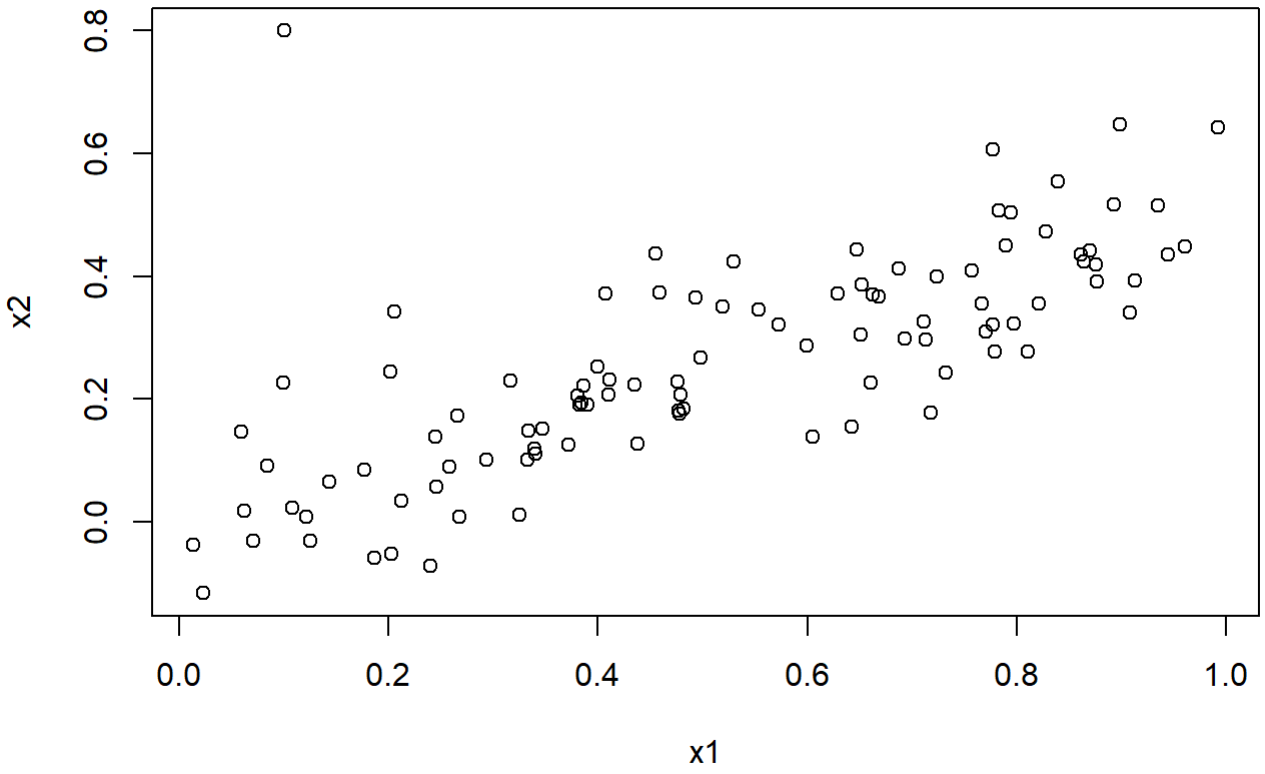
g. Now suppose we obtain one additional observation, which was unfortunately mismeasured.

```
> x1=c(x1, 0.1)
> x2=c(x2, 0.8)
> y=c(y,6)
```

```
x1 = c(x1, 0.1)
x2 = c(x2, 0.8)
y  = c(y,  6.0)
```

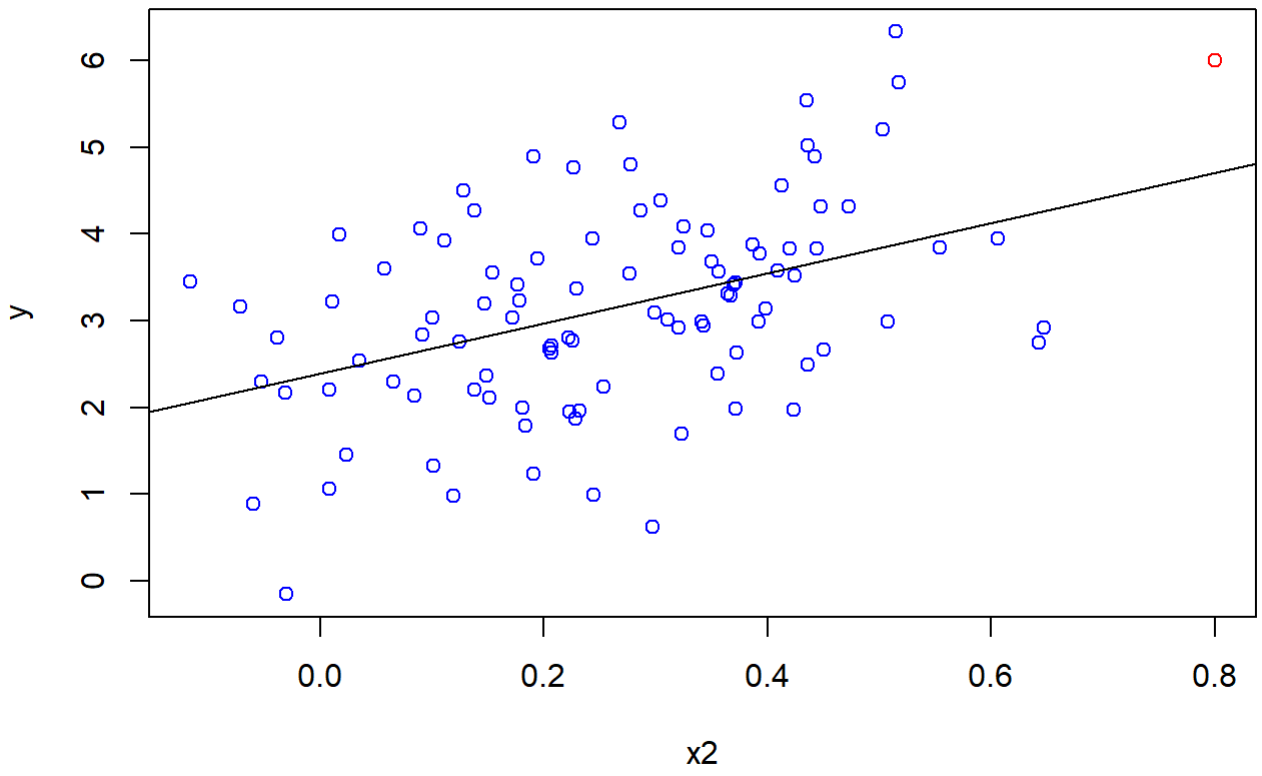
Re-fit the linear models from (c) to (e) using this new data. What effect does this new observation have on each of the models? In each model, is this observation an outlier? A high-leverage point? Both? Explain your answers.

```
plot(x1, x2)
```



From the scatterplot, we can tell that the new observation is definitely a high leverage point, since it does not follow the pattern of all the other points. Also, it does not follow the original equation for the x1-x2 *relationship:* $x2 = \frac{1}{2}x1 + \epsilon$. *Instead of being half the value of* x1, *the value of* x2 *is eight times the value! This either is a very high epsilon or evidence that the original relationship that we had supposed between* x1 *and* x2 *is not actually their true relationship. Deciding which is the case can be very important for producing reliable results.*

```
plot(x2, y, col='blue')
abline(lm.fit2)
points(x2[101], y[101], col='red')
```



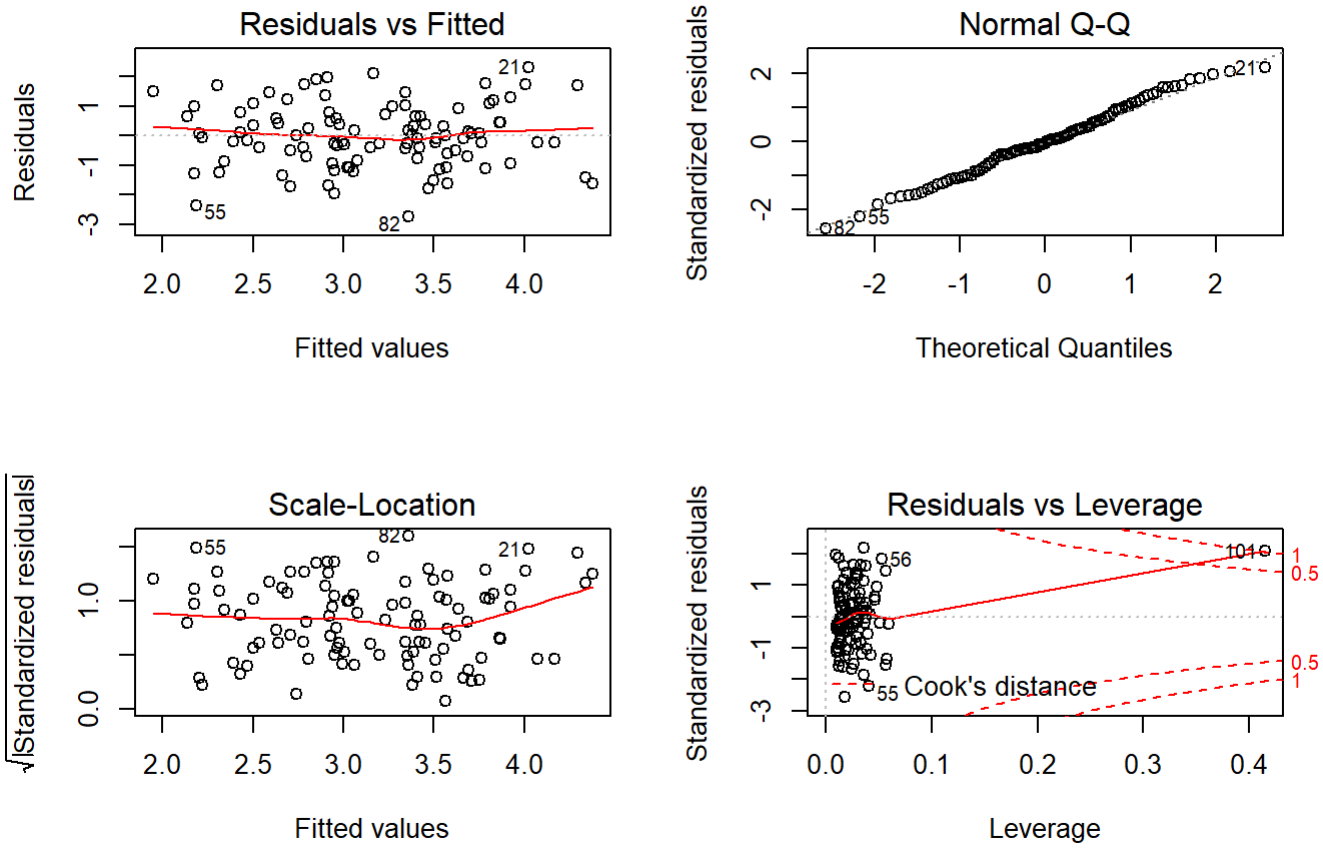
From this plot, the new observation doesn't appear to be much of an outlier.

```
lm.fit3 = lm(y~x1+x2)
summary(lm.fit3)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73348 -0.69318 -0.05263  0.66385  2.30619
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.2267     0.2314   9.624 7.91e-16 ***
## x1            0.5394     0.5922   0.911  0.36458
## x2            2.5146     0.8977   2.801  0.00614 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.075 on 98 degrees of freedom
## Multiple R-squared:  0.2188, Adjusted R-squared:  0.2029
## F-statistic: 13.72 on 2 and 98 DF,  p-value: 5.564e-06
```

Interestingly, the fit using the new observation point now has a low enough p-value to reject the null hypothesis for $\hat{\beta}_2$, but not for $\hat{\beta}_1$. Also, in a fashion similar to the result from part (e), most of the x1 *contribution to the predictor value is being "explained" by the $\hat{\beta}_2$ value.*

```
par(mfrow=c(2,2))
plot(lm.fit3)
```



Looking at these plots, we may reconfirm that the new observation is not an outlier, but it is a high leverage point. This new model does make a good fit to the training data and can be considered close to the true relationship in terms of its ability to predict the response value, but the equation for the model is not actually that close to the true equation used to produce y*.*

Exercise 15

This problem involves the **Boston** dat set, which we saw in the lab for this chapter. We will now try to predict per capita crime rate using the other variables in this data set. In other words, per capita crime rate is the response, and the other variables are the predictors.

```
library(MASS)
?Boston
```

a. For each predictor, fit a simple linear regression model to predict the response. Describe your results. In which of the models is there a statistically significant association between the predictor and the response? Create some plots to back up your assertions.

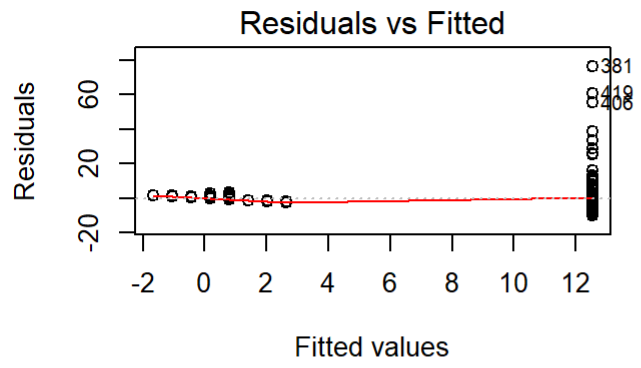
```
for (predictor in names(Boston)) {
  if (predictor != 'crim') {
    print(predictor)
    # Display the p-value for the estimated slope of the fit
    print(coefficients(summary(lm(formula = as.formula(paste("crim ~ ", predictor)), dat
a = Boston)))[2,4])
  }
}
```

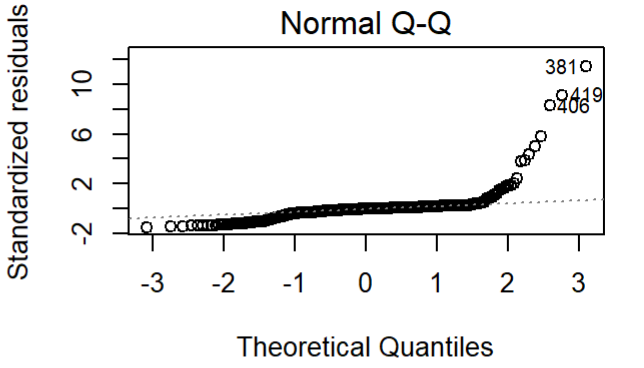
```
## [1] "zn"
## [1] 5.506472e-06
## [1] "indus"
## [1] 1.450349e-21
## [1] "chas"
## [1] 0.2094345
## [1] "nox"
## [1] 3.751739e-23
## [1] "rm"
## [1] 6.346703e-07
## [1] "age"
## [1] 2.854869e-16
## [1] "dis"
## [1] 8.519949e-19
## [1] "rad"
## [1] 2.693844e-56
## [1] "tax"
## [1] 2.357127e-47
## [1] "ptratio"
## [1] 2.942922e-11
## [1] "black"
## [1] 2.487274e-19
## [1] "lstat"
## [1] 2.654277e-27
## [1] "medv"
## [1] 1.173987e-19
```

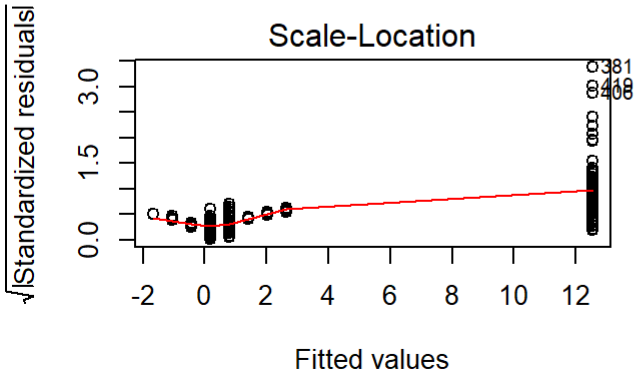
zn, indus, nox, rm, age, dis, rad, tax, ptratio, black, lstat, *and* medv *all have a low enough p-value to reject the null hypothesis for a simple linear regression with* crim. *That is all of the predictors in the* Boston *data except for* chas.

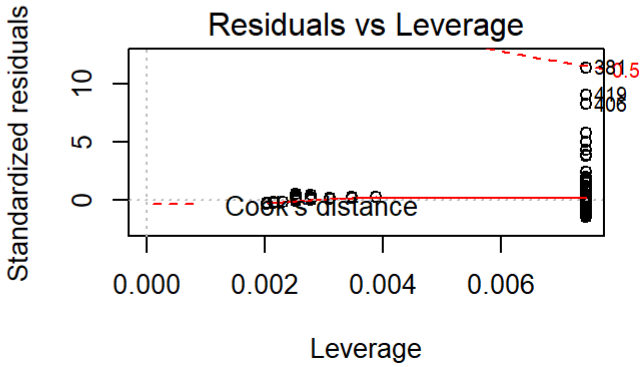
Let's look more closely at the three predictors with the lowest p-values: rad, tax, *and* ptratio.

```
lm.rad = lm(crim ~ rad, data = Boston)
lm.tax = lm(crim ~ tax, data = Boston)
lm.ptratio = lm(crim ~ ptratio, data = Boston)
par(mfrow=c(2,2))
plot(lm.rad)
```

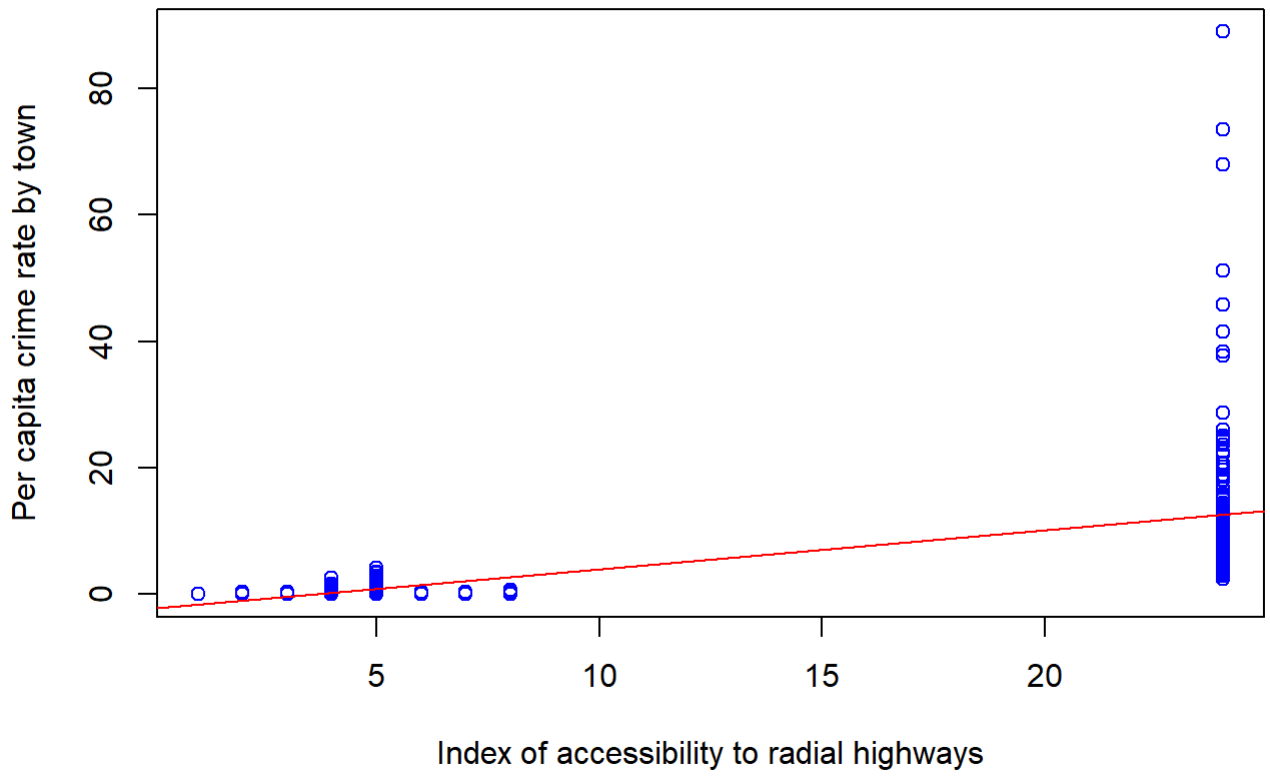






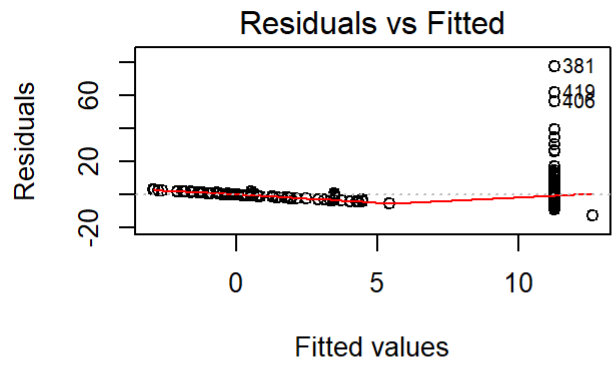


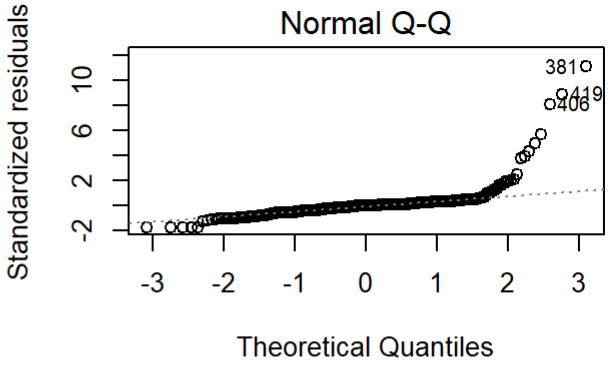

```
plot(Boston$rad, Boston$crim, xlab='Index of accessibility to radial highways', ylab='Pe
r capita crime rate by town', col='blue')
abline(lm.rad, col='red')
```

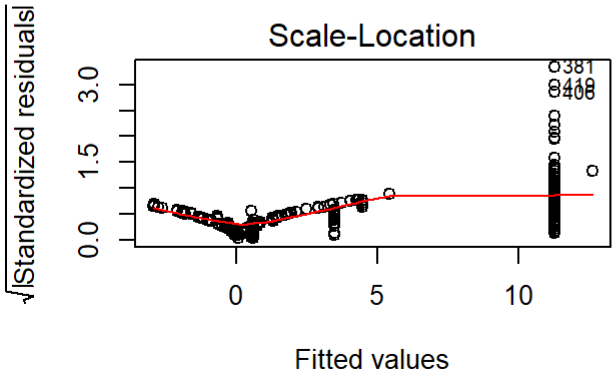


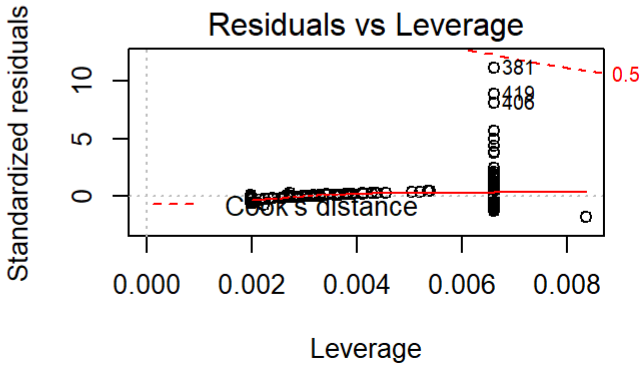
From the plots, it is obvious that a linear regression here is a terrible fit. Understanding the very limited amount of values that rad *can take on also suggests that a linear regression might not be ideal. Despite the low p-value of the fit, we may conclude from our training data that there is not a simple linear relationship between the index of accessibility to radial highways and the per capita crime rate of the town.*

```
par(mfrow=c(2,2))
plot(lm.tax)
```

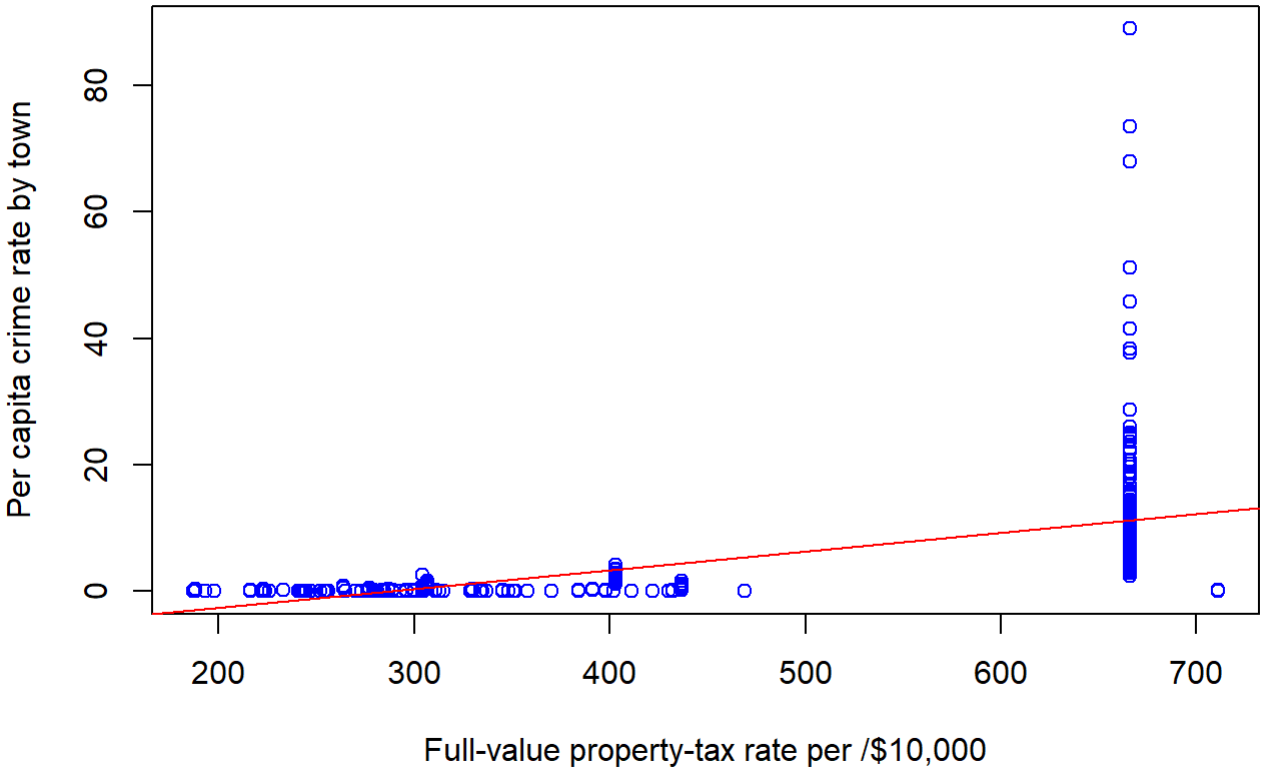






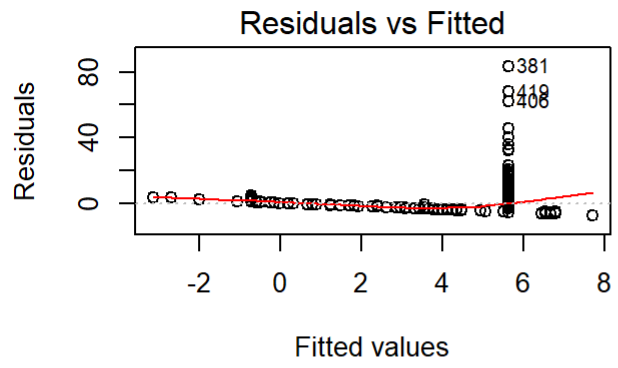


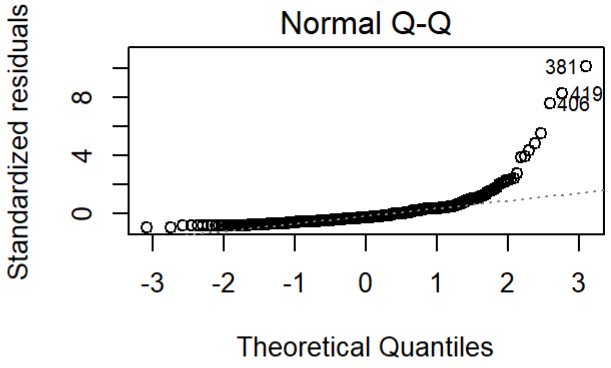

```
plot(Boston$tax, Boston$crim, xlab='Full-value property-tax rate per /$10,000', ylab='Pe
r capita crime rate by town', col='blue')
abline(lm.tax, col='red')
```

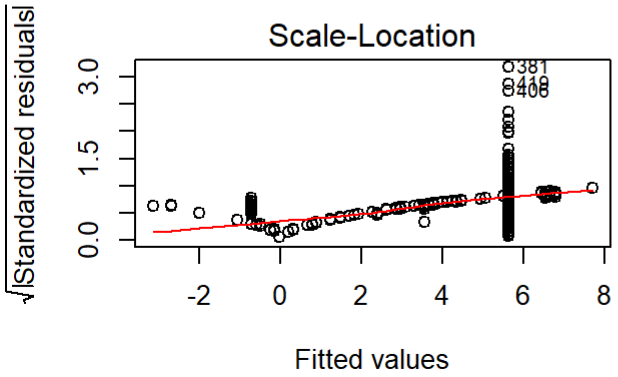


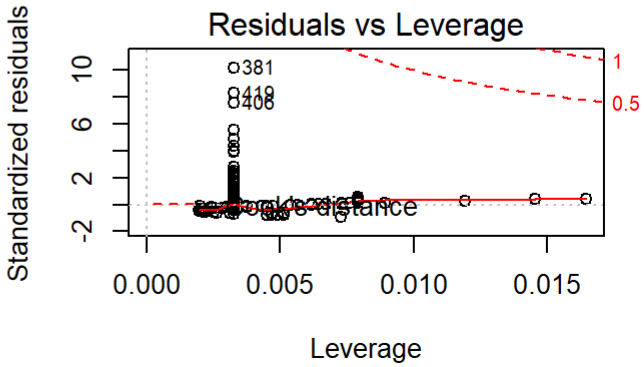
Interestingly, the fit for the crime rate using the property-tax rate is very reminiscent of the fit using the index of accessibility to radial highways. In both fits, the relationship is clearly not linear, but the only towns with high crime rates also have a high predictor value. However, a high predictor value town is still much more likely to have a low per capita crime rate.

```
par(mfrow=c(2,2))
plot(lm.ptratio)
```

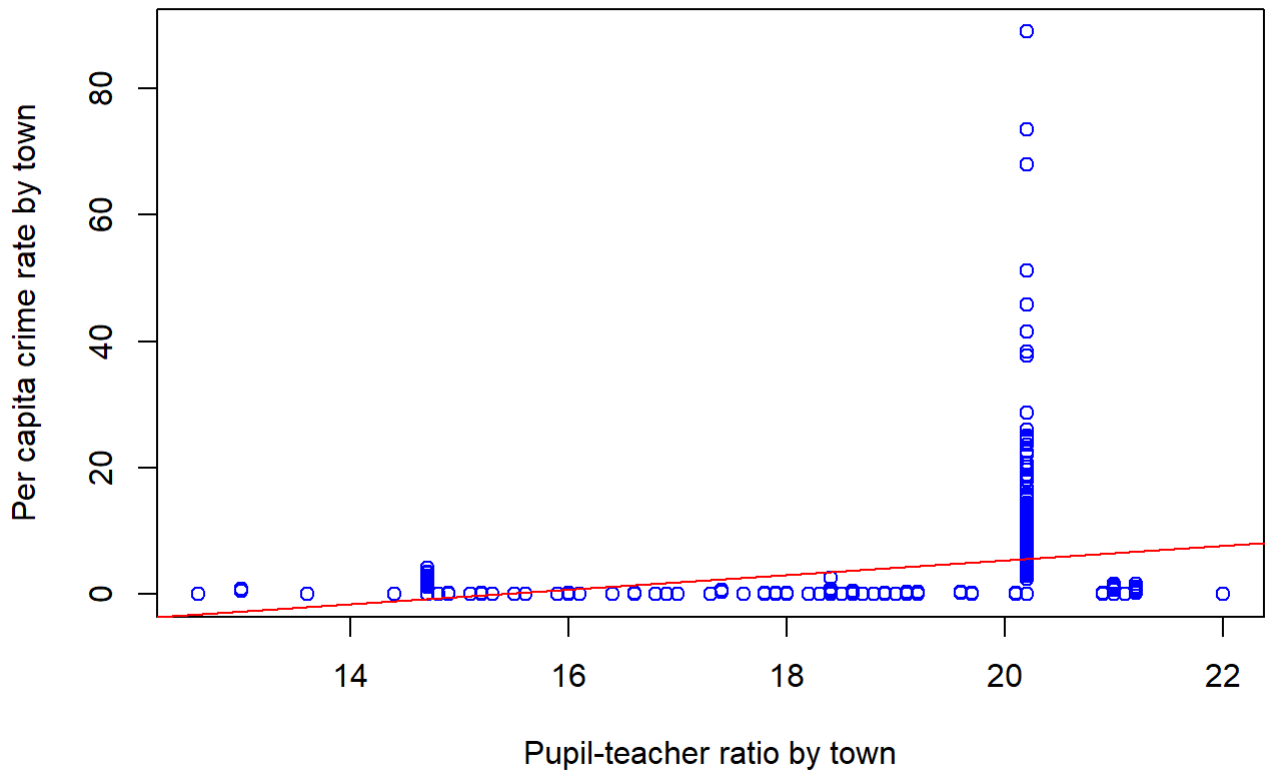









```
plot(Boston$ptratio, Boston$crim, xlab='Pupil-teacher ratio by town', ylab='Per capita c
rime rate by town', col='blue')
abline(lm.ptratio, col='red')
```



Again, we see a similar behavior between the predictor and the response when we use the pupil-teacher ratio. The three predictors with the lowest p-value fits clearly do not have a linear relationship with the response. From these results, we can see that it is very important to consider more than just the p-value when assessing the viability of a fit.

b. Fit a multiple regression model to predict the response using all of the predictors. Describe your results. For which predictors can we reject the null hypothesis $H_0 : \beta_j = 0$?

```
lm.fit = lm(crim ~ ., data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = crim ~ ., data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.924 -2.120 -0.353  1.019 75.051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.033228   7.234903   2.354 0.018949 *
## zn            0.044855   0.018734   2.394 0.017025 *
## indus        -0.063855   0.083407  -0.766 0.444294
## chas         -0.749134   1.180147  -0.635 0.525867
## nox         -10.313535   5.275536  -1.955 0.051152 .
## rm            0.430131   0.612830   0.702 0.483089
## age           0.001452   0.017925   0.081 0.935488
## dis          -0.987176   0.281817  -3.503 0.000502 ***
## rad           0.588209   0.088049   6.680 6.46e-11 ***
## tax          -0.003780   0.005156  -0.733 0.463793
## ptratio      -0.271081   0.186450  -1.454 0.146611
## black        -0.007538   0.003673  -2.052 0.040702 *
## lstat         0.126211   0.075725   1.667 0.096208 .
## medv         -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454,  Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
```

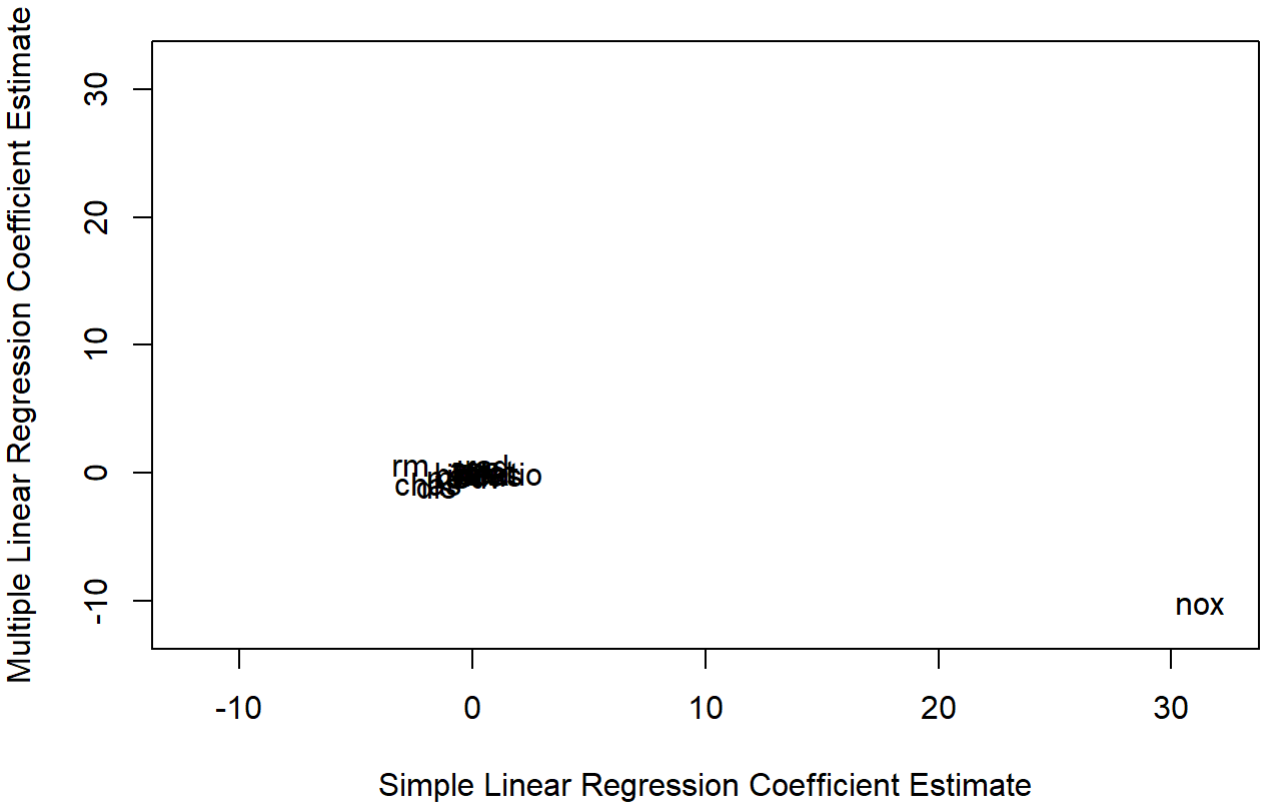
Using a threshold of 1%, we may reject the null hypothesis for the dis, rad, *and* medv *predictors. This is a much smaller subset of predictors than we saw when using simple linear regressions. Also, of the three predictors with the lowest p-values from the simple linear regressions, only one may reject the null hypothesis here.*

c. How do your results from (a) compare to your results from (b)? Create a plot displaying the univariate regression coefficients from (a) on the *x*-axis and the multiple regression coefficients from (b) on the *y*-axis. That is, each predictor is displayed as a single point in the plot. Its coefficient in a simple linear regression model is shown on the *x*-axis, and its coefficient estimate in the multiple linear regression model is shown on the *y*-axis.

```
df = data.frame()
# Drop the first name b/c that's the Intercept
for (name in names(coef(lm.fit))[-1]){
  df = rbind(df, data.frame(name = name,
                            simple = coef(lm(formula = as.formula(paste("crim ~ ", nam
e)), data = Boston))[name],
                            multiple = coef(lm.fit)[name]
                            ))
}
print(df)
```

```
##            name      simple      multiple
## zn           zn -0.07393498   0.044855215
## indus     indus  0.50977633  -0.063854824
## chas       chas -1.89277655  -0.749133611
## nox         nox 31.24853120 -10.313534912
## rm           rm -2.68405122   0.430130506
## age         age  0.10778623   0.001451643
## dis         dis -1.55090168  -0.987175726
## rad         rad  0.61791093   0.588208591
## tax         tax  0.02974225  -0.003780016
## ptratio ptratio  1.15198279  -0.271080558
## black     black -0.03627964  -0.007537505
## lstat     lstat  0.54880478   0.126211376
## medv       medv -0.36315992  -0.198886821
```

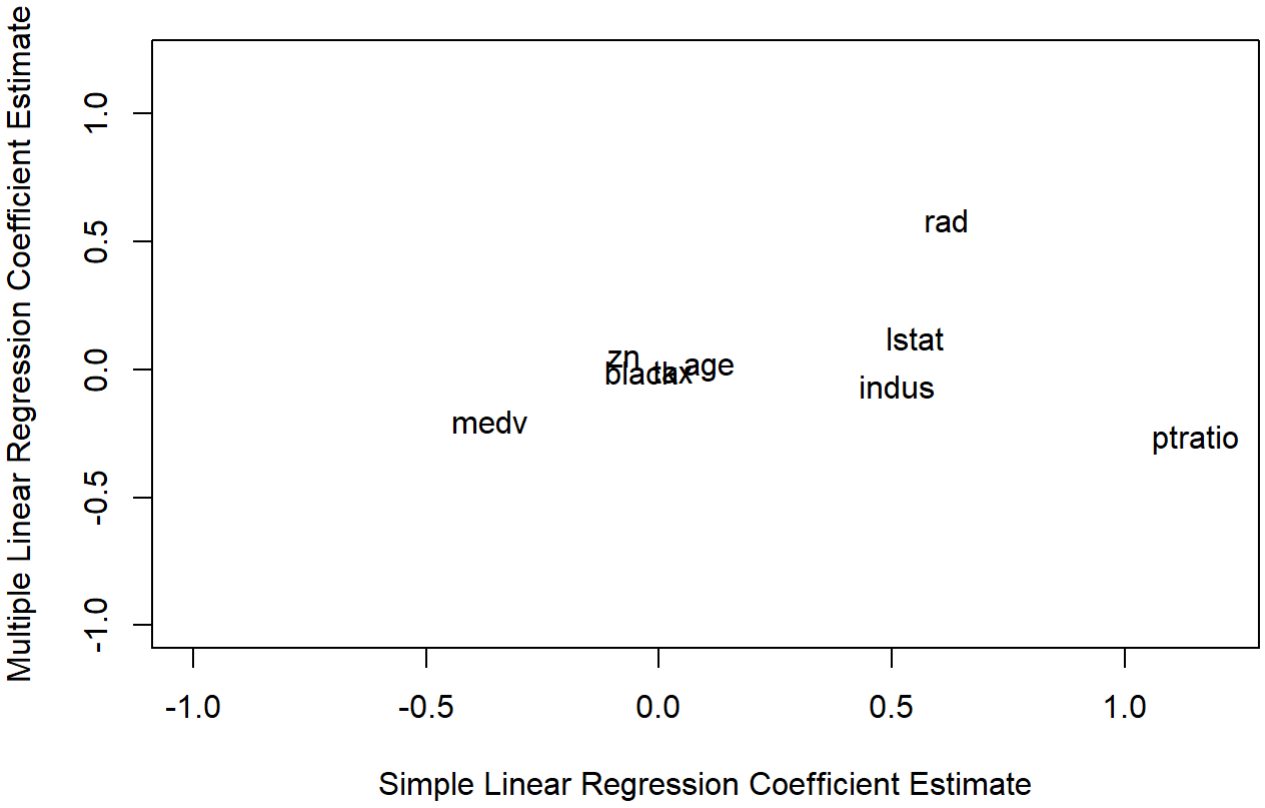
```
plot(1, type='n', xlab = 'Simple Linear Regression Coefficient Estimate', ylab = 'Multip
le Linear Regression Coefficient Estimate', xlim=c(-12,32), ylim=c(-12,32))
text(df$simple, df$multiple, df$name)
```




```
#plot(df$simple, df$multiple, xlab = 'Simple Linear Regression Coefficient Estimate', yl
ab = 'Multiple Linear Regression Coefficient Estimate', xlim=c(-12,32), ylim=c(-12,32))
```

The estimated coefficient for the nox *predictor changed dramatically. In the simple linear regression, the model had an* **increase** *of 31 per capita crime rate in a town if it had one more part per 10 million of nitrogen oxides concentration. However, in the multiple linear regression, the estimated impact of one more part per 10 million is a* **decrease** *of 10 per capita crime rate in the town.*

```
subDf = df[which(df$name != 'nox'),]
plot(1, type='n', xlab = 'Simple Linear Regression Coefficient Estimate', ylab = 'Multip
le Linear Regression Coefficient Estimate', xlim=c(-1,1.2), ylim=c(-1,1.2))
text(subDf$simple, subDf$multiple, subDf$name)
```



None of the other coefficient estimates change quite as dramatically from the simple linear regression to the multiple linear regression. However, the zn, indus, rm, tax, *and* ptratio *predictors all switch signs, either going from a negative slope to a positive slope or from a positive slope to a negative slope. Looking just at the* tax *and* ptratio *predictors, they both went from a positive estimated coefficient with a very low p-value in their simple linear regressions to a negative estimated coefficient with a very high p-value. These changes demonstrate just how important it is to carefully select what type of model to implement.*

d. Is there evidence of non-linear association between any of the predictors and the response? To answer this question, for each predictor X, fit a model of the form

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$$

.

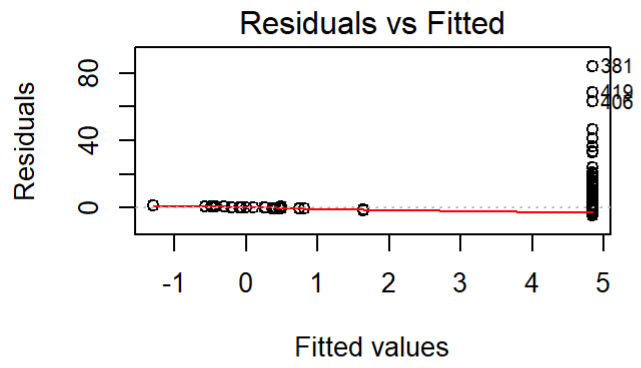
```
for (predictor in names(Boston)) {
  if (predictor != 'crim') {
    print(predictor)
    fit = lm(formula = as.formula(paste('crim ~ I(', predictor, '^3) + I(', predictor,
'^2) + ', predictor)), data = Boston)
    # Display the p-values for the estimated slope of the fit
    print(coef(summary(fit)))
    print('XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX')
  }
}
```

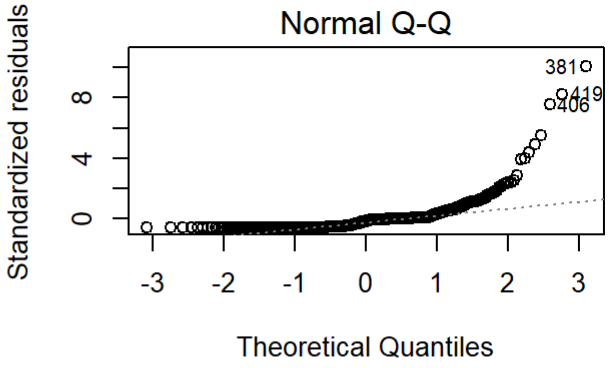
```
## [1] "zn"
##                  Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)  4.846050e+00 4.329846e-01 11.192199 4.113000e-26
## I(zn^3)     -3.775793e-05 3.138615e-05 -1.203012 2.295386e-01
## I(zn^2)      6.482634e-03 3.860728e-03  1.679122 9.375050e-02
## zn          -3.321884e-01 1.098081e-01 -3.025171 2.612296e-03
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "indus"
##                 Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)  3.662568279 1.5739833325  2.326942 2.036526e-02
## I(indus^3)  -0.006976009 0.0009566596 -7.292049 1.196405e-12
## I(indus^2)   0.251937298 0.0393221245  6.407011 3.420187e-10
## indus       -1.965212934 0.4819900578 -4.077289 5.297064e-05
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "chas"
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  3.744447  0.3961111  9.453021 1.239505e-19
## I(chas^3)   -1.892777  1.5061155 -1.256727 2.094345e-01
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "nox"
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)   233.0866    33.6431  6.928214 1.311939e-11
## I(nox^3)    -1245.7029   149.2816 -8.344649 6.961110e-16
## I(nox^2)     2248.5441   279.8993  8.033404 6.811300e-15
## nox         -1279.3713   170.3975 -7.508159 2.758372e-13
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "rm"
##               Estimate Std. Error    t value   Pr(>|t|)
## (Intercept) 112.624596  64.517236  1.7456513 0.08148335
## I(rm^3)      -0.174477   0.263747 -0.6615314 0.50857511
## I(rm^2)       4.550896   5.009862  0.9083875 0.36410939
## rm          -39.150136  31.311492 -1.2503440 0.21175641
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "age"
##                  Estimate   Std. Error    t value    Pr(>|t|)
## (Intercept) -2.548763e+00 2.769140e+00 -0.9204168 0.357797077
## I(age^3)     5.745307e-05 2.109355e-05  2.7237266 0.006679915
## I(age^2)    -7.229596e-03 3.636995e-03 -1.9877935 0.047377328
## age          2.736531e-01 1.863796e-01  1.4682566 0.142660827
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "dis"
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  30.0476116  2.4458697 12.285042 1.633088e-30
## I(dis^3)     -0.1185986  0.0204004 -5.813544 1.088832e-08
## I(dis^2)      2.4520722  0.3464194  7.078333 4.941214e-12
## dis         -15.5543535  1.7359674 -8.960049 6.374792e-18
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "rad"
##                  Estimate Std. Error    t value  Pr(>|t|)
## (Intercept) -0.605544746 2.05010768 -0.2953722 0.7678317
## I(rad^3)     0.003208996 0.00456401  0.7031090 0.4823138
## I(rad^2)    -0.075177363 0.14854303 -0.5060982 0.6130099
## rad          0.512736038 1.04359675  0.4913162 0.6234175
```

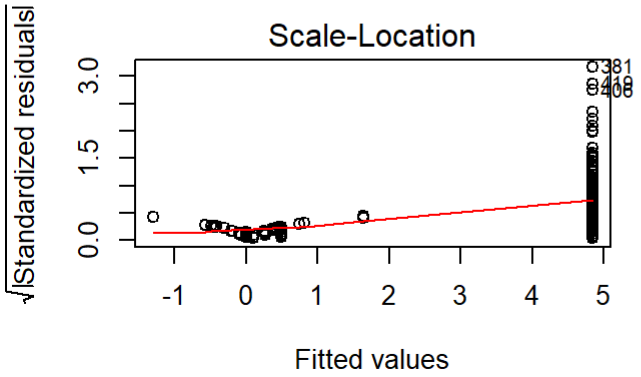
```
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "tax"
##                  Estimate   Std. Error   t value  Pr(>|t|)
## (Intercept)  1.918358e+01 1.179555e+01  1.626341 0.1045047
## I(tax^3)    -2.203715e-07 1.888705e-07 -1.166786 0.2438507
## I(tax^2)     3.608266e-04 2.425463e-04  1.487661 0.1374682
## tax         -1.533096e-01 9.567806e-02 -1.602349 0.1097075
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "ptratio"
##                  Estimate   Std. Error   t value    Pr(>|t|)
## (Intercept)  477.18404610 156.79497750  3.043363 0.002462124
## I(ptratio^3)  -0.08476032   0.03089749 -2.743275 0.006300514
## I(ptratio^2)   4.63534723   1.60832115  2.882103 0.004119552
## ptratio      -82.36053772  27.64394247 -2.979334 0.003028663
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "black"
##                  Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)  1.826370e+01 2.304903e+00  7.9238468 1.497097e-14
## I(black^3)  -2.652453e-07 4.364264e-07 -0.6077665 5.436172e-01
## I(black^2)   2.137404e-04 2.984193e-04  0.7162418 4.741751e-01
## black       -8.355805e-02 5.632751e-02 -1.4834323 1.385871e-01
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "lstat"
##                  Estimate   Std. Error    t value   Pr(>|t|)
## (Intercept)  1.2009655812 2.0286451580  0.5920038 0.55411459
## I(lstat^3)  -0.0008573703 0.0005651667 -1.5170220 0.12989059
## I(lstat^2)   0.0557794159 0.0301156072  1.8521764 0.06458736
## lstat       -0.4490655872 0.4648911092 -0.9659586 0.33452999
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
## [1] "medv"
##                 Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) 53.165538094 3.3563105351  15.840471 3.989088e-46
## I(medv^3)   -0.001490103 0.0002037895  -7.311969 1.046510e-12
## I(medv^2)    0.155496490 0.0171904422   9.045520 3.260523e-18
## medv        -5.094830543 0.4338320525 -11.743785 2.637707e-28
## [1] "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

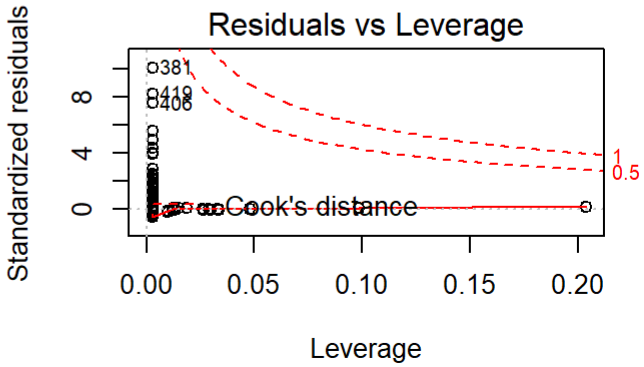
zn, indus, nox, dis, *and* medv *all have low p-values which indicate that a polynomial fit could be a good model.*

```
lm.zn = lm(crim ~ I(zn^3) + I(zn^2) + zn, data = Boston)
lm.indus = lm(crim ~ I(indus^3) + I(indus^2) + indus, data = Boston)
lm.nox = lm(crim ~ I(nox^3) + I(nox^2) + nox, data = Boston)
lm.dis= lm(crim ~ I(dis^3) + I(dis^2) + dis, data = Boston)
lm.medv = lm(crim ~ I(medv^3) + I(medv^2) + medv, data = Boston)
par(mfrow = c(2,2))
plot(lm.zn)
```

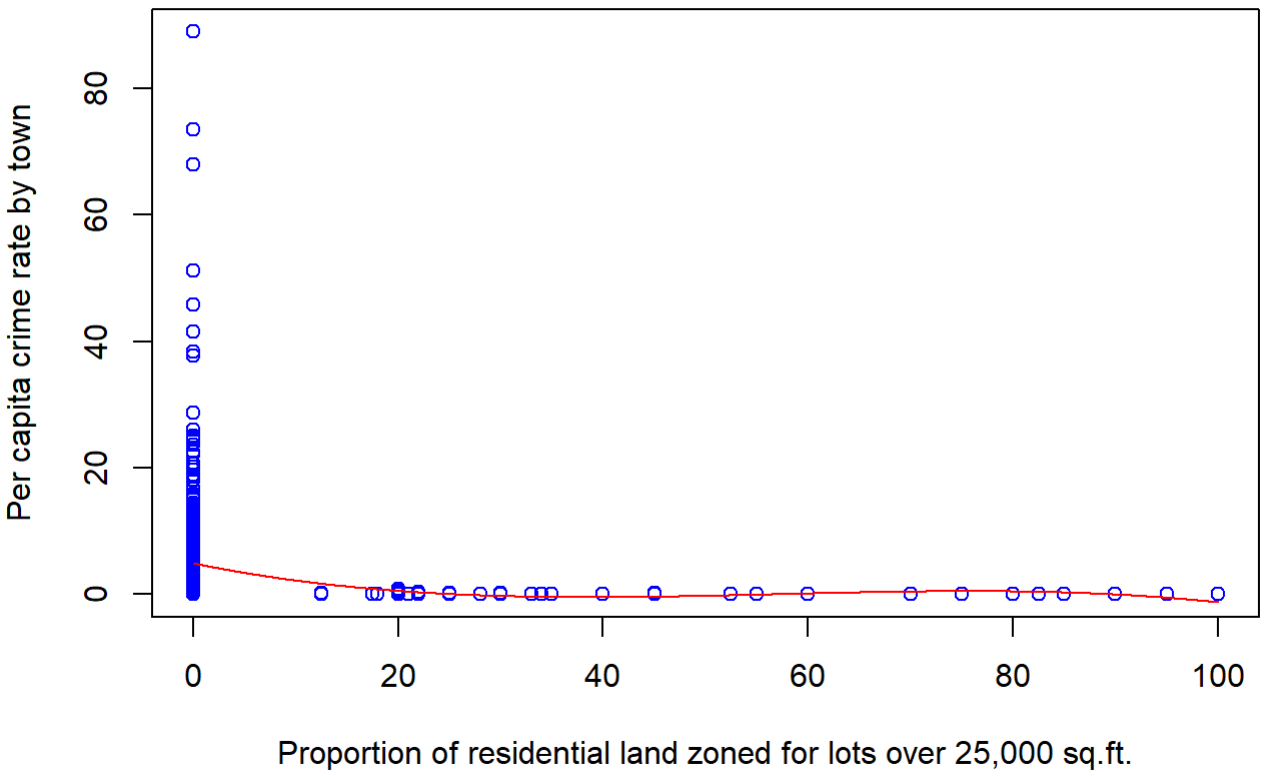




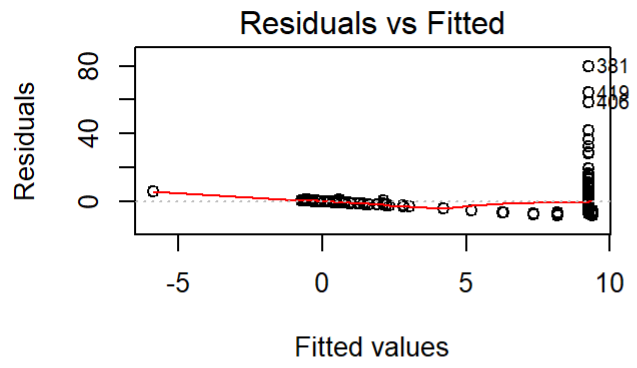


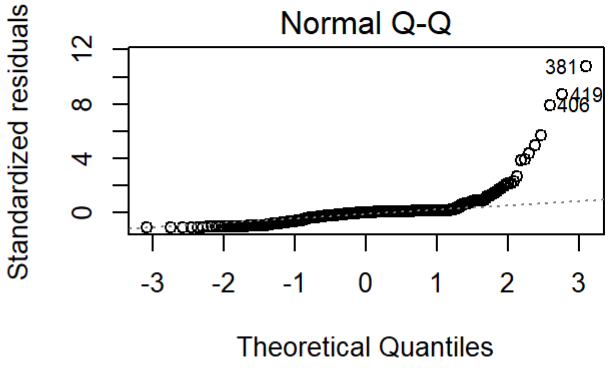


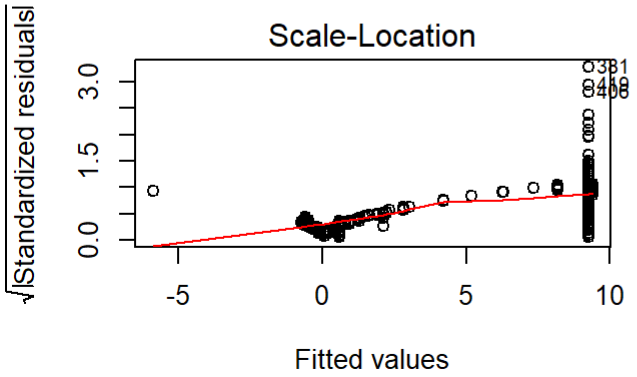

```
plot(Boston$zn, Boston$crim, xlab = 'Proportion of residential land zoned for lots over
 25,000 sq.ft.', ylab = 'Per capita crime rate by town', col='blue')
x = seq(from = min(Boston$zn), to = max(Boston$zn), length.out = 1000)
lines(x, predict(lm.zn, data.frame(zn = x)), col='red')
```

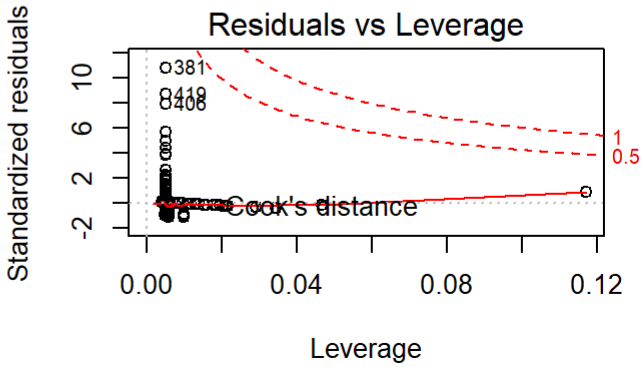



```
par(mfrow = c(2,2))
plot(lm.indus)
```

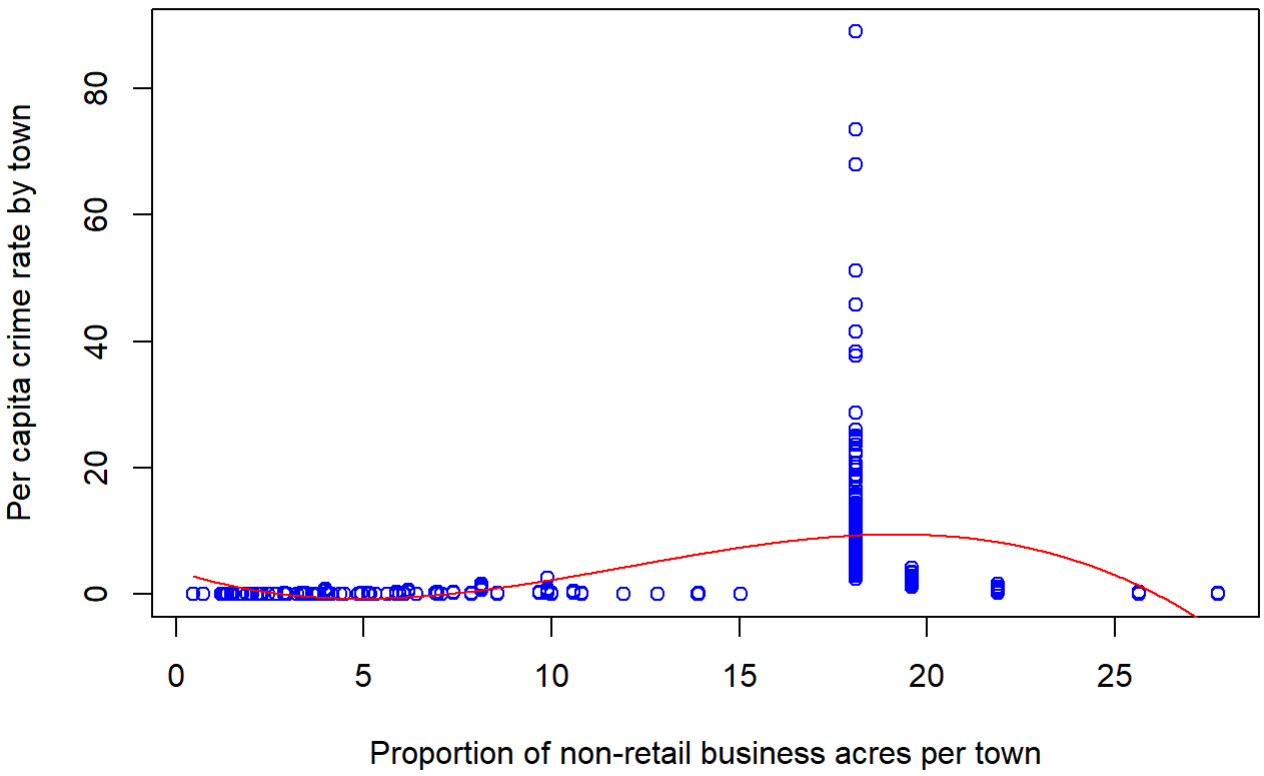




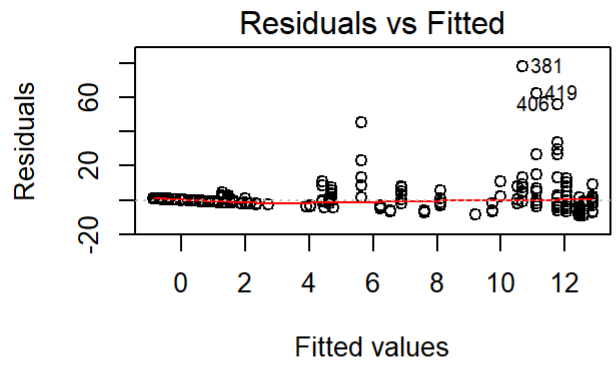


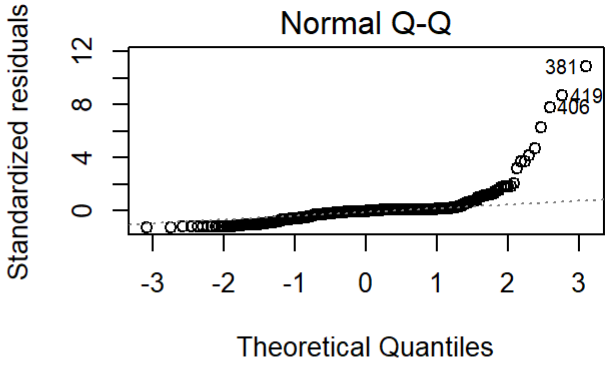


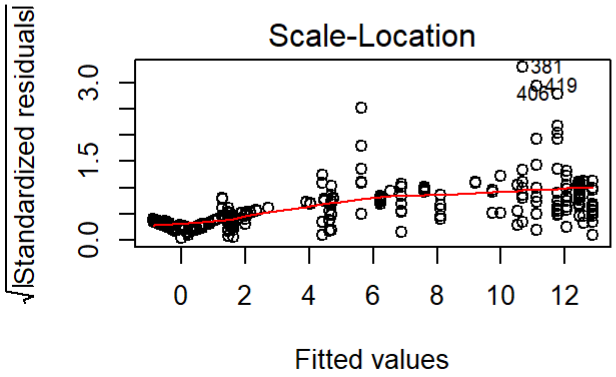

```
plot(Boston$indus, Boston$crim, xlab = 'Proportion of non-retail business acres per tow
n', ylab = 'Per capita crime rate by town', col='blue')
x = seq(from = min(Boston$indus), to = max(Boston$indus), length.out = 1000)
lines(x, predict(lm.indus, data.frame(indus = x)), col='red')
```

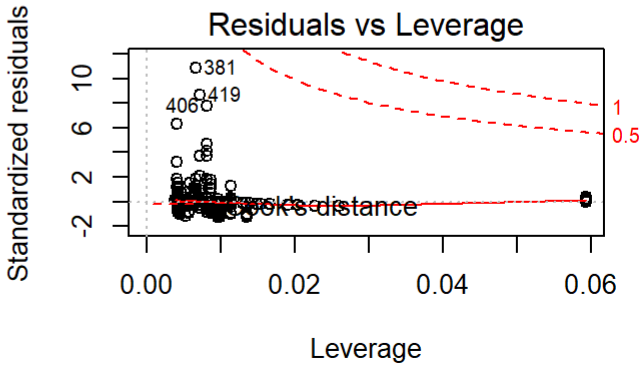



```
par(mfrow = c(2,2))
plot(lm.nox)
```

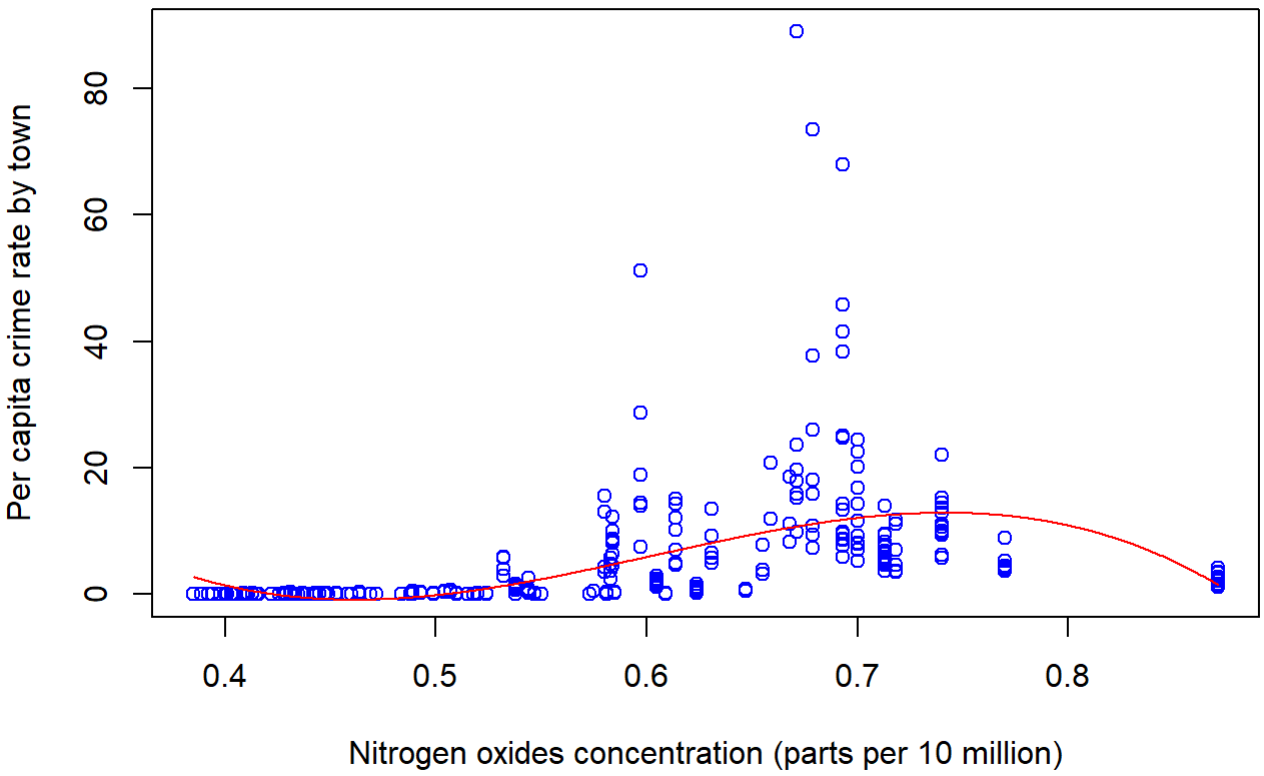




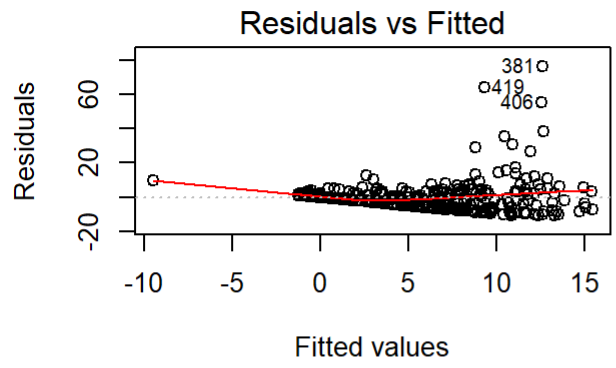


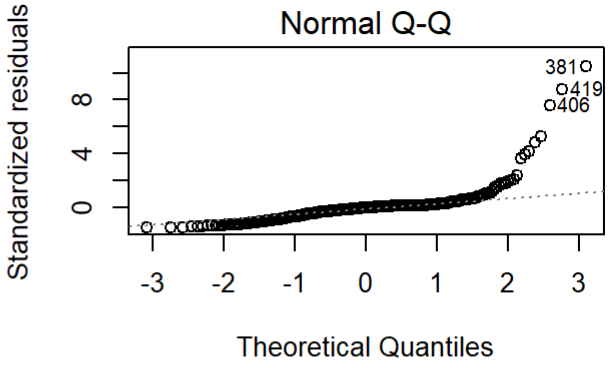


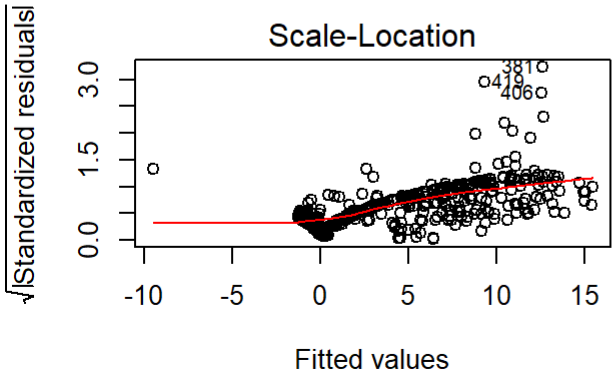

```
plot(Boston$nox, Boston$crim, xlab = 'Nitrogen oxides concentration (parts per 10 millio
n)', ylab = 'Per capita crime rate by town', col='blue')
x = seq(from = min(Boston$nox), to = max(Boston$nox), length.out = 1000)
lines(x, predict(lm.nox, data.frame(nox = x)), col='red')
```

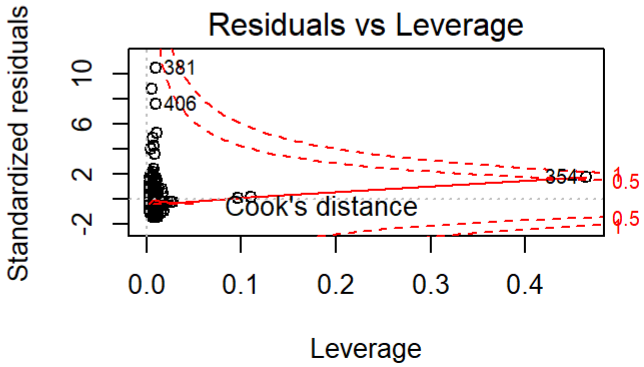



```
par(mfrow = c(2,2))
plot(lm.dis)
```

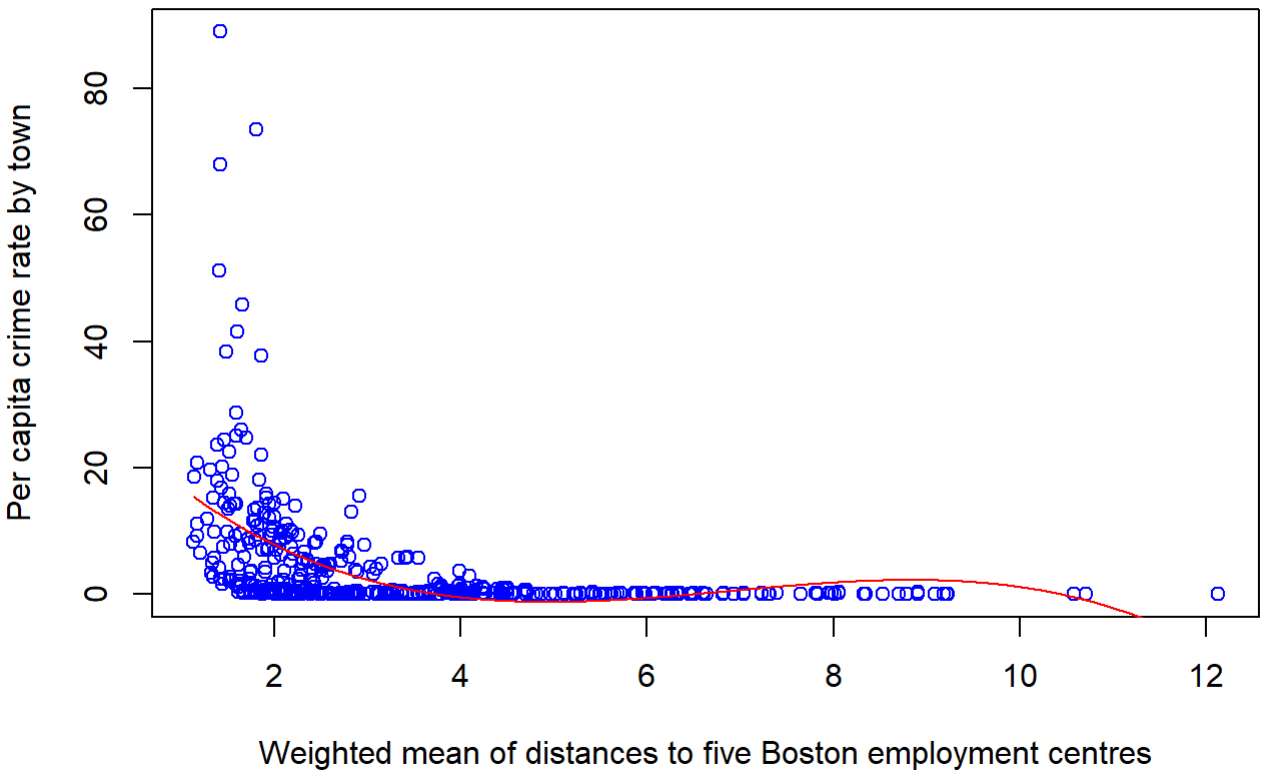




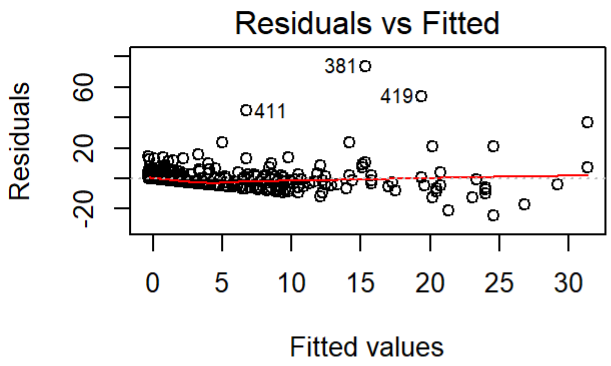


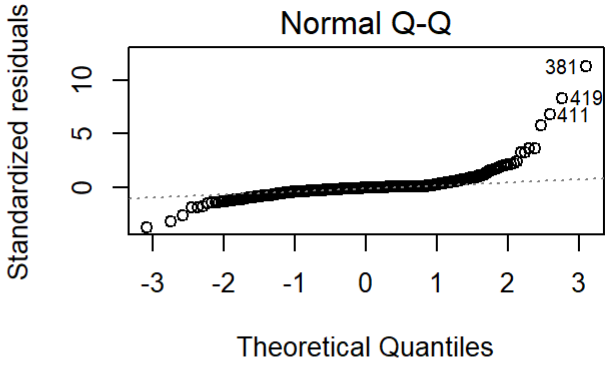


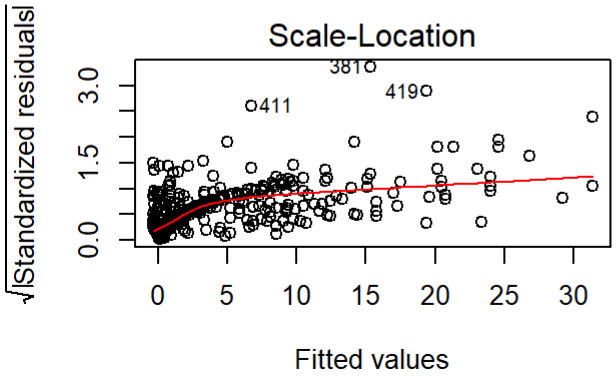

```
plot(Boston$dis, Boston$crim, xlab = 'Weighted mean of distances to five Boston employme
nt centres', ylab = 'Per capita crime rate by town', col='blue')
x = seq(from = min(Boston$dis), to = max(Boston$dis), length.out = 1000)
lines(x, predict(lm.dis, data.frame(dis = x)), col='red')
```

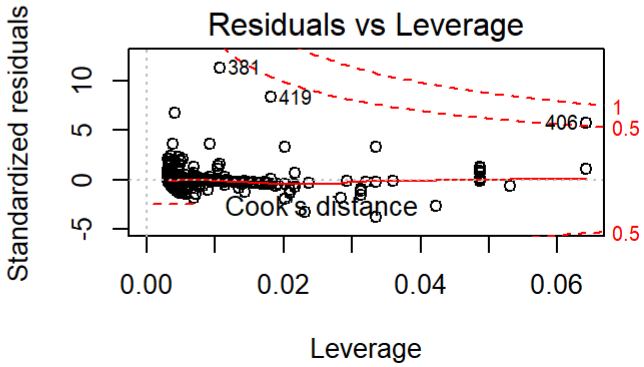



```
par(mfrow = c(2,2))
plot(lm.medv)
```

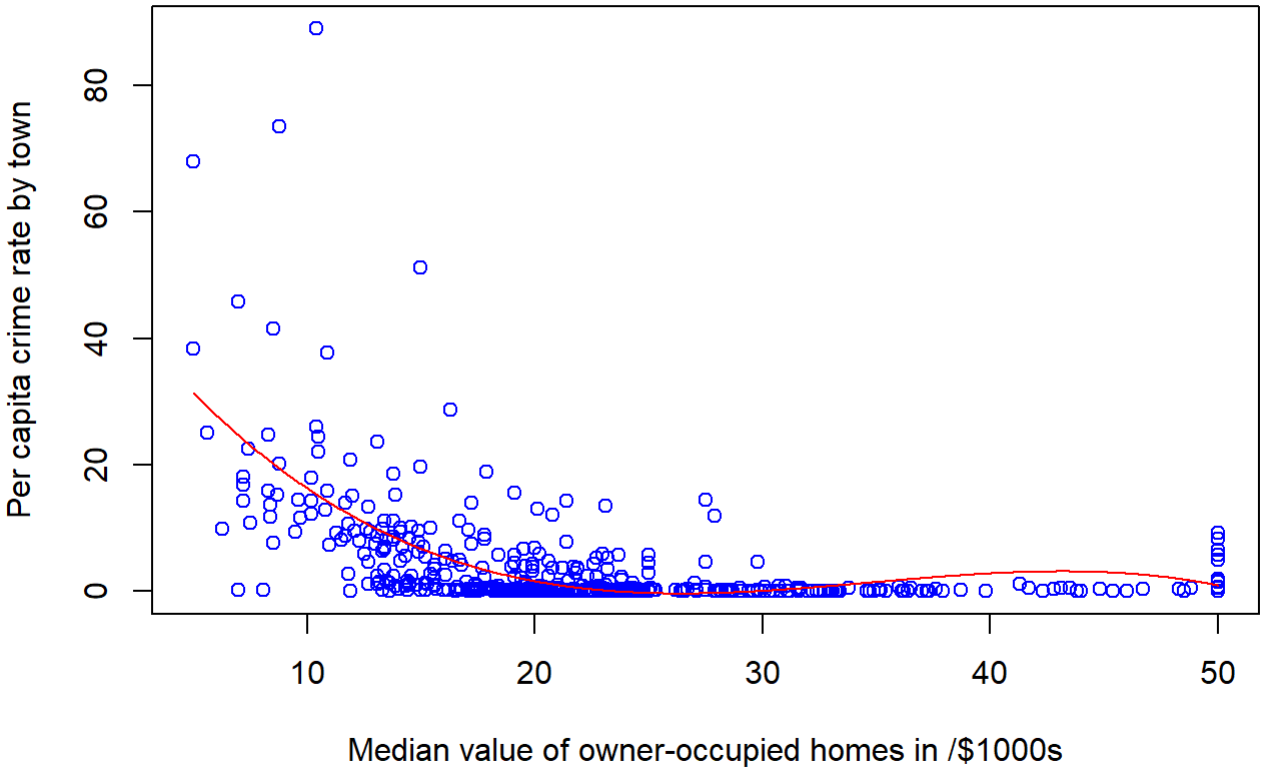









```
plot(Boston$medv, Boston$crim, xlab = 'Median value of owner-occupied homes in /$1000s',
ylab = 'Per capita crime rate by town', col='blue')
x = seq(from = min(Boston$medv), to = max(Boston$medv), length.out = 1000)
lines(x, predict(lm.medv, data.frame(medv = x)), col='red')
```



The dis *and* medv *predictors seem like they actually do have a close-to-polynomical relationship with the* crim *response. The* zn *and* indus *predictors don't seem to have a polynomial relationship with the* crim *response. The* nox *predictor could have a relationship with the* crim *response, but it doesn't seem like it. We would need more observations to say for sure.*