512: Assignment #2

Brody Vogel

2/2/2018

Preparation

```r
set.seed(1234)
library(ISLR)
```

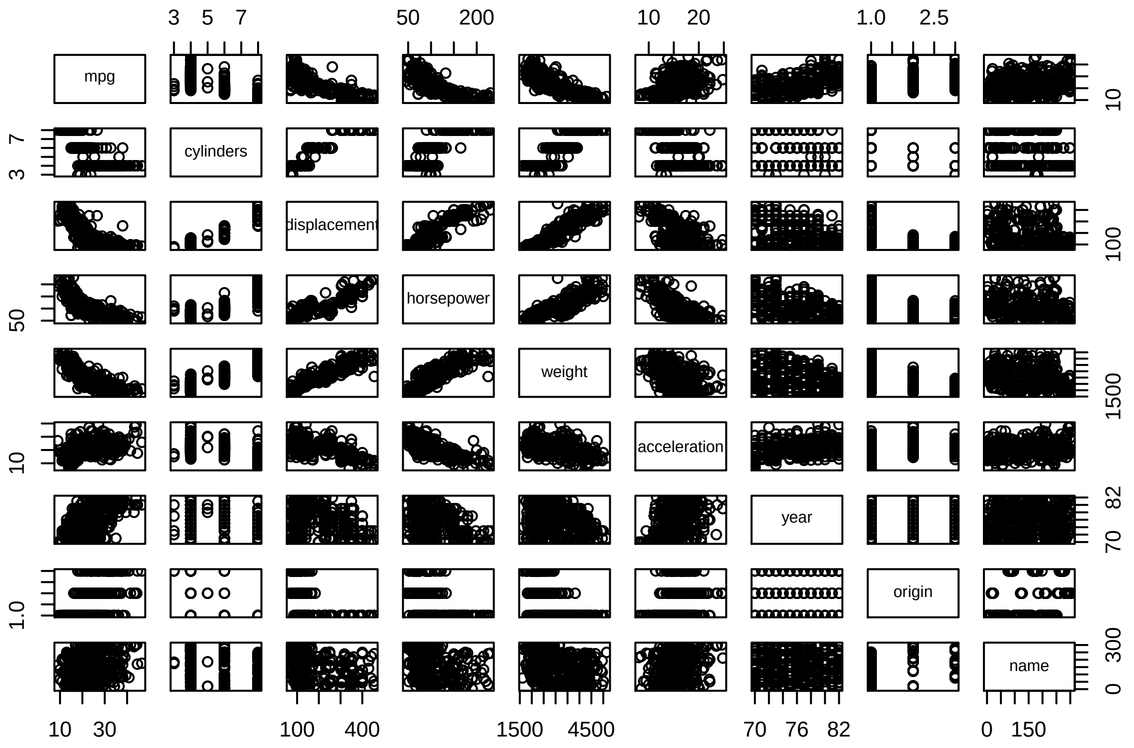
Problem 1 (#4)

a) We'd expect the training RSS to be lower for the cubic regression, because it has the predictor from the linear fit (X) plus two more (X^2, X^3). With these extra predictors, the cubic model could better adjust to the irreducible error in the training data and so reduce the training RSS.
b) We'd expect the test RSS to be much lower for the linear regression model, because the cubic model would overfit to the training data. If the true relationship is linear, the cubic model would have too much variance, i.e., it would adjust too much to the irreducible error in the training data and so perform poorly on new test data.
c) Again, we'd expect the training RSS to be lower for the cubic regression, also becaue it has the predictor from the linear fit (X) plus two more (X^2, X^3). To reiterate, these extra predictors would allow the cubic model to better adjust to the irreducible error in the training data and so reduce the training RSS. With respect to training RSS, it doesn't much matter what the *true* relationship between response and predictor(s) is; models with more predictors will always reduce the training RSS because of their added flexibility.
d) There isn't enough information in the question to be certain. On the one hand, if the true relationship *is not* far from linear, the linear regression will certainly produce a lower test RSS for the same reasons mentioned in (b) - that is, a cubic regression would overfit to the training data. But, on the other hand, if the true relationship *is* far from linear, we'd expect the cubic regression to have a far lower test RSS, because the linear regression would be too biased for a relationship of this complexity.

Problem 2 (#9 a-c)

a)

```r
data(Auto)
pairs(Auto)
```



b)

```r
cor(subset(Auto, select = -name))
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration       year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
## acceleration    1.0000000  0.2903161  0.2127458
## year            0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

c)

```r
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
## 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) Yes, it appears that there is a relationship between the predictors and the response. The R^2 of $\approx$ .82 - although sometimes misleading - is a good indication that the model is capturing a lot of the variance in the data. Better evidence, though, is the F-statistic and it's p-value of essentially 0. This is almost a sure sign that there is a relationship between the reponse and predictors.

.ii) The predictors that test as having a statistically-significant relationship to the resposne are: Displacement, Weight, Year, and Origin, as evidenced by their low p-values.

.iii) The coefficient for the year variable ($\approx$ .75) suggests that, for every year newer that a car is, it gets $\approx$ .75 more miles per gallon than a car one year older. That is, for every year in the data set, cars get about .75 more miles per gallon.

Problem 3 (#12)

a) In this situation, $\hat{\beta}_{y->x} = \hat{\beta}_{x->y} <=> \frac{\sum_{i=1}^{n} x_i \times y_i}{\sum_{i'=1}^{n} x_{i'}^2} = \frac{\sum_{i=1}^{n} y_i \times x_i}{\sum_{i'=1}^{n} y_{i'}^2}$. Because the top terms are equal, this is equivalent to the situation in which $\sum_{i'=1}^{n} x_{i'}^2 = \sum_{i'=1}^{n} y_{i'}^2$, or, in other words, the situation in which the sum of squares of y is the same as the sum of squares of x.

b) Below, $\hat{\beta}_{y->x} \approx 3.13 \neq \hat{\beta}_{x->y} \approx .28$, and so the coefficient estimates are different based on the predictor/response assignments of the variables.

```r
set.seed(1)
x <- rnorm(100)
y <- 1 + 3*x

lm.fit2 <- lm(y~x-1)
lm.fit3 <- lm(x~y-1)

summary(lm.fit2)$coef
```

```
##   Estimate Std. Error  t value     Pr(>|t|)
## x 3.134337  0.1108134 28.28481 3.161789e-49
```

```
summary(lm.fit3)$coef
```

```
##    Estimate Std. Error  t value     Pr(>|t|)
## y 0.2839137 0.01003767 28.28481 3.161789e-49
```

c) Below, $\hat{\beta}_{y->x} = -1 = \hat{\beta}_{x->y}$, and so the coefficient estimates are the same regardless of the predictor/response assignments of the variables.

```
x <- rnorm(100)
y <- -x

lm.fit4 <- lm(y~x-1)
lm.fit5 <- lm(x~y-1)

summary(lm.fit4)$coef
```

```
## Warning in summary.lm(lm.fit4): essentially perfect fit: summary may be
## unreliable
```

```
##   Estimate   Std. Error       t value Pr(>|t|)
## x       -1 6.051834e-17 -1.652392e+16        0
```

```
summary(lm.fit5)$coef
```

```
## Warning in summary.lm(lm.fit5): essentially perfect fit: summary may be
## unreliable
```

```
##   Estimate   Std. Error       t value Pr(>|t|)
## y       -1 6.051834e-17 -1.652392e+16        0
```

Problem 4 (#13)

a)

```
set.seed(1)

x <- rnorm(100)
```

b)

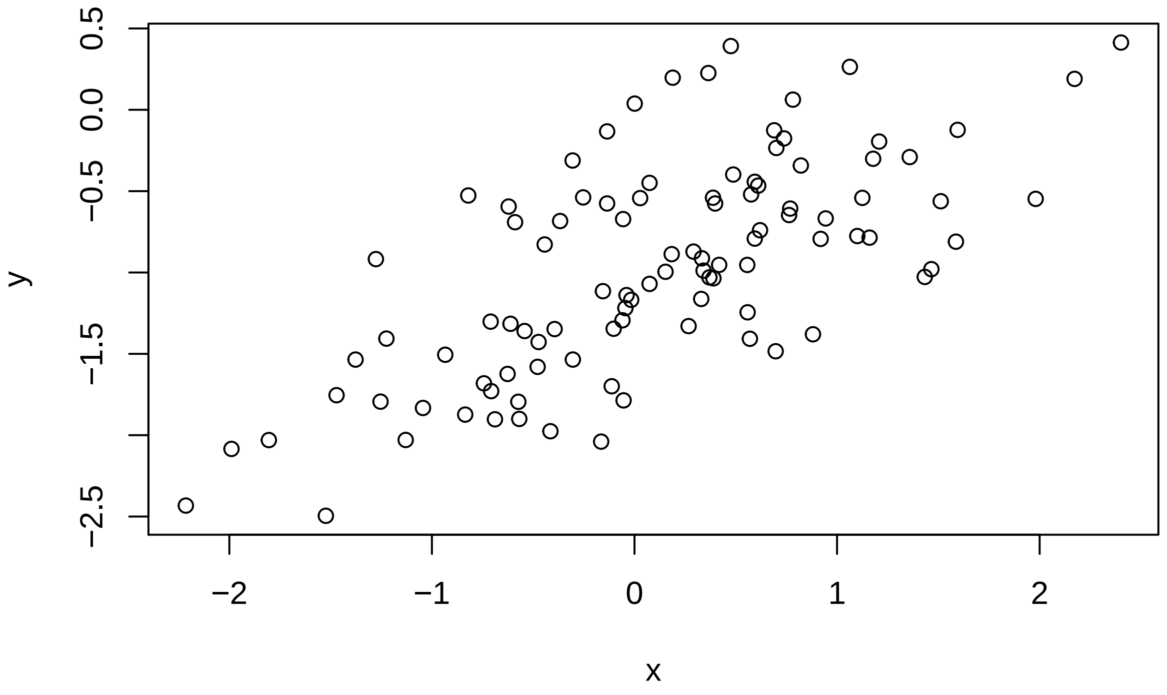
```
eps <- rnorm(100, 0, sqrt(.25))
```

c) The length of y is 100; $\beta_0 = -1$; $\beta_1 = .5$.

```
y <- -1 + .5*x + eps
```

d) The relationship between x and y looks broadly linear, with a positive slope and a fair bit of variance.

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



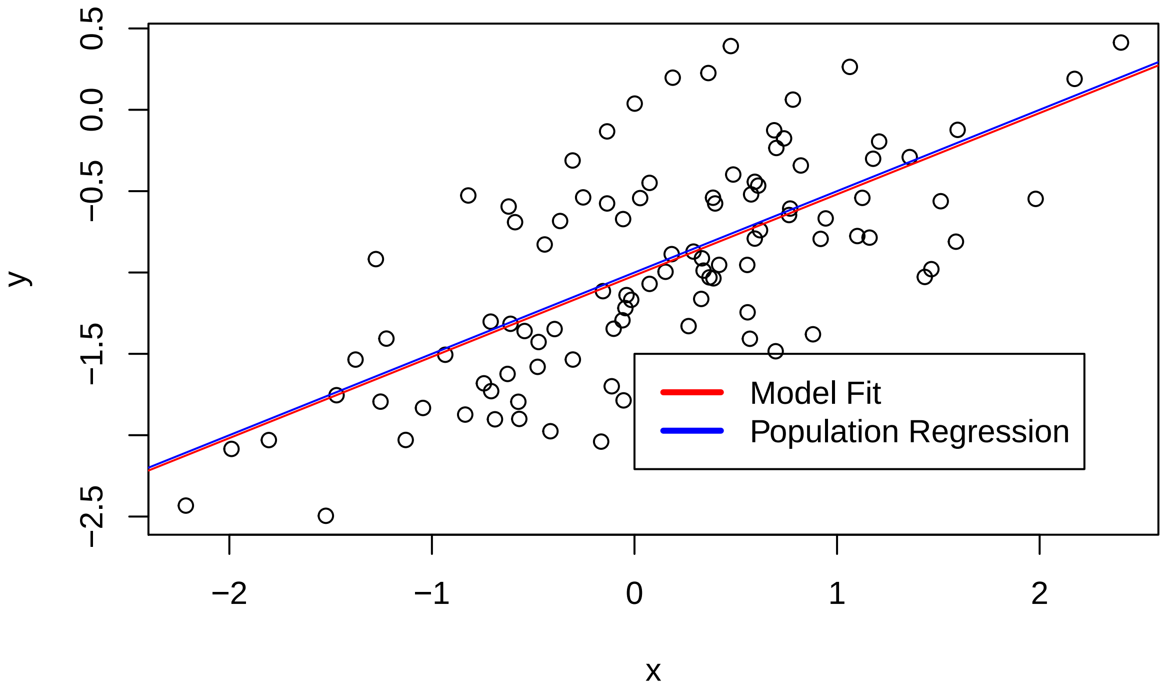
e) The model is about what we'd expect, given the apparent linear nature of the data. That is, the simple linear regression fit's large F-statistic tells us the response and predictor are almost certainly related. The coefficients of the model are $\hat{\beta}_0 \approx -1.01, \hat{\beta}_1 \approx .49$, which are very close to the true $\beta_0 = -1, \beta_1 = .5$.

```
lm.fit6 <- lm(y~x)
summary(lm.fit6)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93842 -0.30688 -0.06975  0.26970  1.17309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
## x            0.49947    0.05386   9.273 4.58e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 98 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4619
## F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

f)

```
plot(x, y)

abline(lm.fit6, col = 'red')
abline(-1, .5, col = 'blue')

legend(.0, -1.5, c('Model Fit', 'Population Regression'), col = c('red', 'blue'), lwd = 3)
```



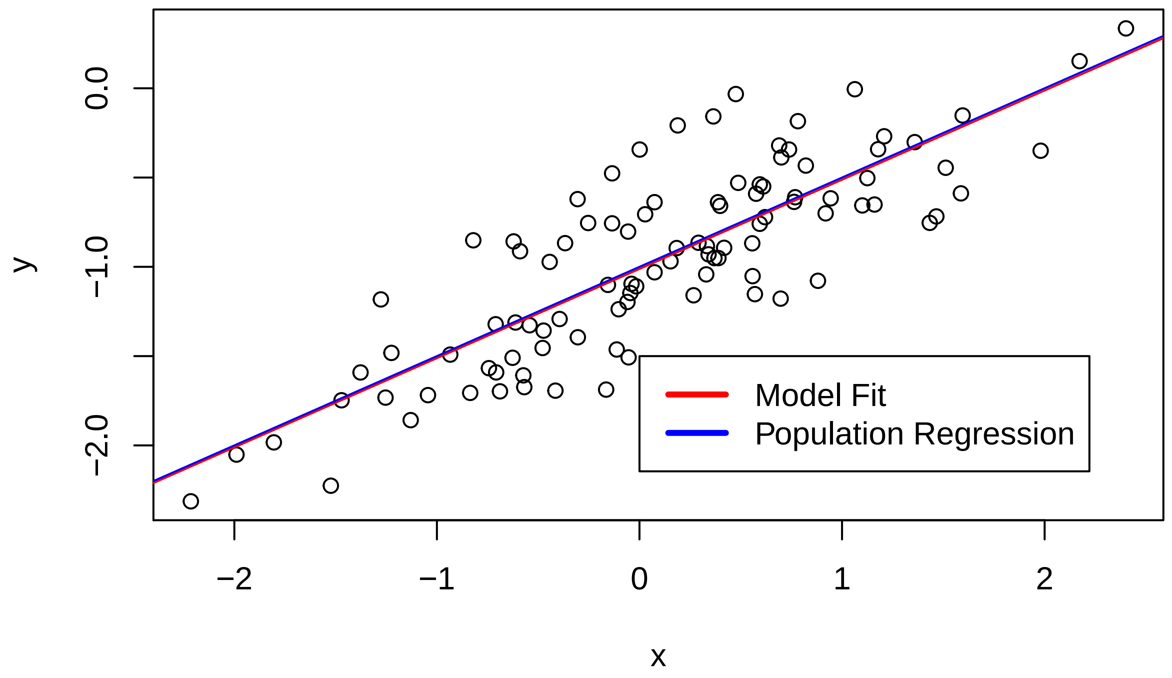
g) There is the slimmest of evidence that the quadratic term improves the fit of the model, but the improvement is almost certainly not enough to justify the added variance of a higher-order term. On the one had, the RSE decreases from .4814 to .479, and the R^2 increases from .4674 to .4779, which is technically evidence of an improvement. But, for one, these improvements are very small and so indicate adding the quadratic term does little to the model's predictions (also, adding the second predictor will *always* improve the R^2, so such a miniscule improvement is more of a sign that the added predictor is not really significant). And, more importantly, the low p-value of the quadratic term suggests it's not significant in the model.

```
lm.fit7 <- lm(y~x+I(x^2))
summary(lm.fit7)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98252 -0.31270 -0.06441  0.29014  1.13500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.97164    0.05883 -16.517  < 2e-16 ***
## x            0.50858    0.05399   9.420  2.4e-15 ***
## I(x^2)      -0.05946    0.04238  -1.403    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.479 on 97 degrees of freedom
## Multiple R-squared:  0.4779, Adjusted R-squared:  0.4672
## F-statistic:  44.4 on 2 and 97 DF,  p-value: 2.038e-14
```

h) For this model with less noise, the fit of the model improves because of the decreased variance in the data, as evidenced by the lower RSE, higher R^2, and convergence of the model and population regression lines in the plot. By reducing the noise by over half as I did here, the RSE dipped from .4814 to .3044, the R^2 jumped from .4674 to .687, and the red line that represents the model fit basically overlapped the blue line representing the true population regression.

```r
set.seed(1)

x <- rnorm(100)
eps <- rnorm(100, 0, sqrt(.10))
y <- -1 + .5*x + eps

lm.fit8 <- lm(y~x)

plot(x, y)
abline(lm.fit8, col = 'red')
abline(-1, .5, col = 'blue')
legend(0, -1.5, c('Model Fit', 'Population Regression'), col = c('red', 'blue'), lwd = 3)
```



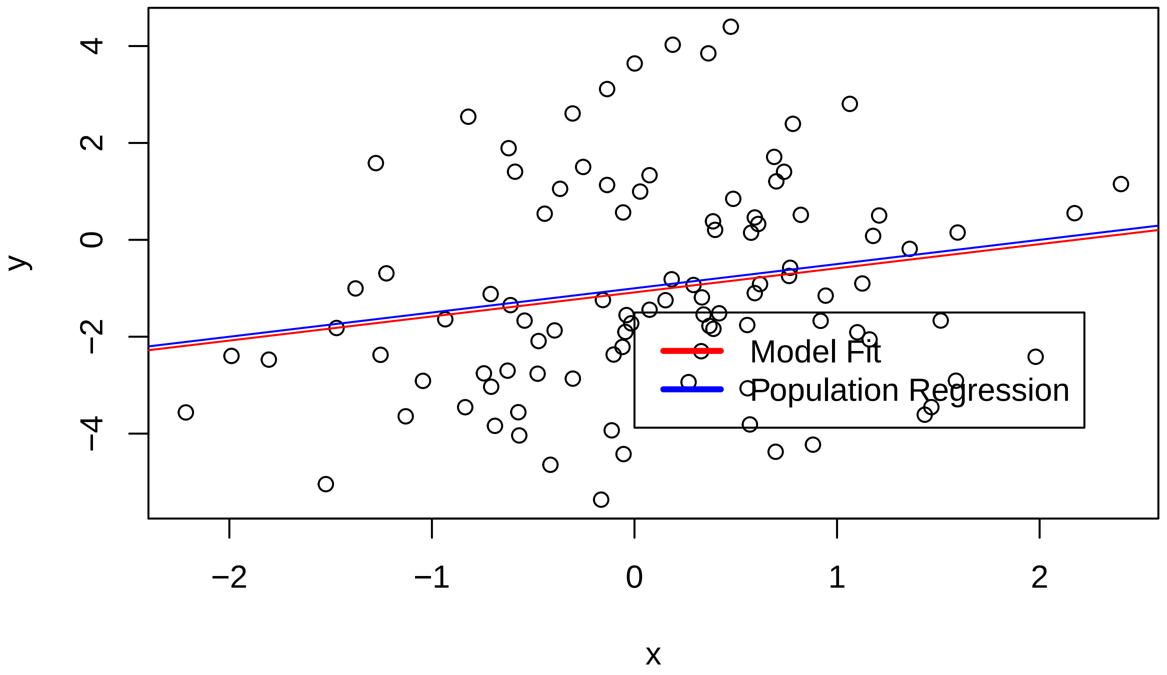
```r
summary(lm.fit8)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.59351 -0.19409 -0.04411  0.17057  0.74193
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01192    0.03067  -32.99   <2e-16 ***
## x            0.49966    0.03407   14.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.3044 on 98 degrees of freedom
## Multiple R-squared:  0.687,  Adjusted R-squared:  0.6838
## F-statistic: 215.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

i) For this model with more noise, the fit of the model worsens because of the increased variance in the data, as evidenced by the higher RSE, lower R^2, and divergence of the model and population regression lines in the plot. By increasing the noise as I did here, the RSE jumped from .4814 to .6808, the R^2 dipped from .4674 to .3047, and the red line that represents the model fit separated itself significantly from the blue line representing the true population regression.

```r
set.seed(1)

x <- rnorm(100)
eps <- rnorm(100, 0, sqrt(5))
y <- -1 + .5*x + eps

lm.fit9 <- lm(y~x)

plot(x, y)
abline(lm.fit9, col = 'red')
abline(-1, .5, col = 'blue')
legend(0, -1.5, c('Model Fit', 'Population Regression'), col = c('red', 'blue'), lwd = 3)
```



```r
summary(lm.fit9)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1967 -1.3724 -0.3119  1.2061  5.2462
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.0843     0.2169  -5.000 2.52e-06 ***
## x             0.4976     0.2409   2.066   0.0415 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.153 on 98 degrees of freedom
## Multiple R-squared:  0.04173,    Adjusted R-squared:  0.03195
## F-statistic: 4.268 on 1 and 98 DF,  p-value: 0.04148
```

j) For β_0, each interval seems to be centered around -1, which is what we'd expect given the true relationship between the response and predictor. With less noise, the interval gets tighter; with more noise, it spreads out a bit.

For β_1, each interval seems to be centered around .5, which is what we'd expect given the true relationship between the response and predictor. With less noise, the interval gets a bit tighter; with more noise, it spreads out pretty significantly.

In sum, then, more noise - i.e., more variance in the data - results in wider confidence intervals for the coefficients. In essence, this means that with more irreducible error in the model, we should be less confident in our models and their predictions.

```r
confint(lm.fit6)
```

```
##                  2.5 %     97.5 %
## (Intercept) -1.1150804 -0.9226122
## x            0.3925794  0.6063602
```

```r
confint(lm.fit8)
```

```
##                  2.5 %     97.5 %
## (Intercept) -1.0727832 -0.9510557
## x            0.4320613  0.5672681
```

```r
confint(lm.fit9)
```

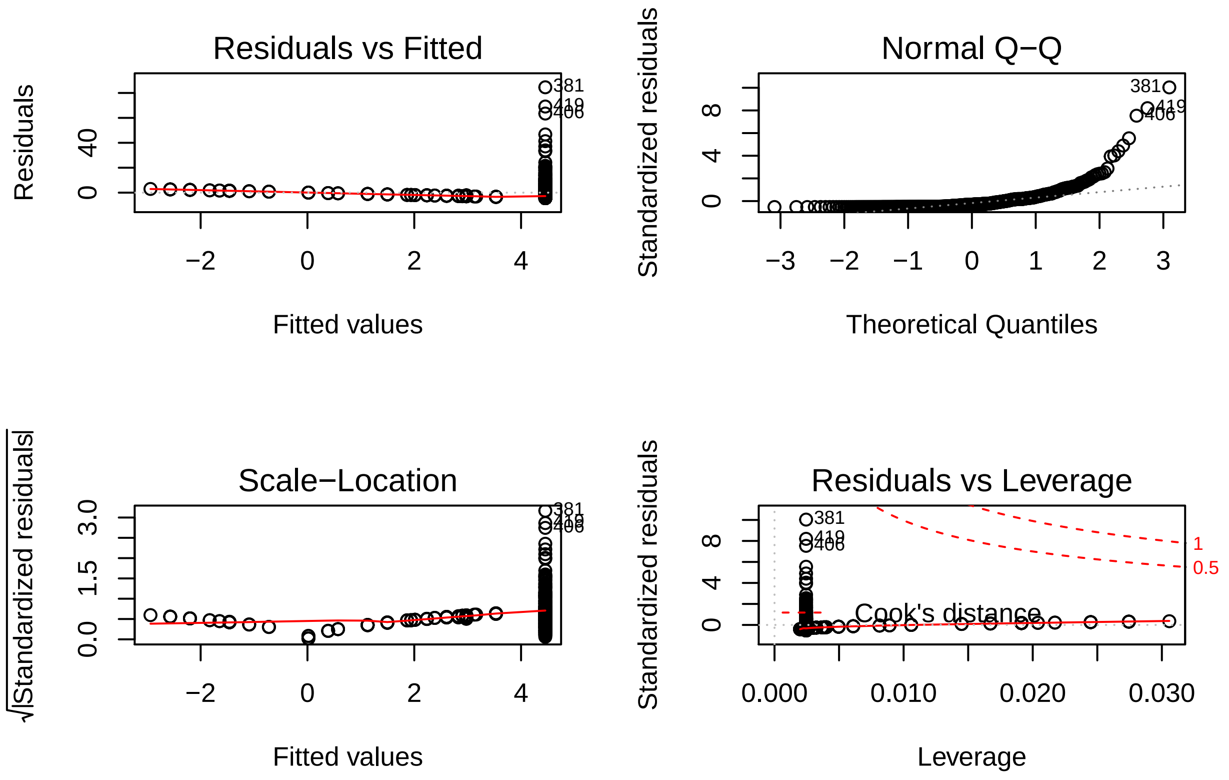
```
##                   2.5 %     97.5 %
## (Intercept) -1.51465507 -0.6539114
## x            0.01960053  0.9756573
```

Problem 5 (#15 a-c)

a) From the summaries, each predictor tests as having a statistically significant relationship with the response, *crim*, except for *chas*. The residual plots seem to be in-line with that statement, but they also make it clear that there are a lot more factors at play than the simple linear model can account for. For example, there seems to be something happening in what appears to be zone 5 in the plot of lm(crim~zn), as the model produces significantly larger residuals at that point.

```r
library(MASS)
data(Boston)

B1 <- lm(crim~zn, data = Boston)
B2 <- lm(crim~indus, data = Boston)
B3 <- lm(crim~chas, data = Boston)
B4 <- lm(crim~nox, data = Boston)
B5 <- lm(crim~rm, data = Boston)
```

```r
B6 <- lm(crim~age, data = Boston)
B7 <- lm(crim~dis, data = Boston)
B8 <- lm(crim~rad, data = Boston)
B9 <- lm(crim~tax, data = Boston)
B10 <- lm(crim~ptratio, data = Boston)
B11 <- lm(crim~black, data = Boston)
B12 <- lm(crim~lstat, data = Boston)
B13 <- lm(crim~medv, data = Boston)

#summary(B1)
#summary(B2)
#...
#summary(B13)

par(mfrow = c(2,2))
plot(B1)
```



```r
#plot(B2)
#...
#plot(B13)
par(mfrow = c(1,1))
```

b) Not nearly as many predictors came out as significant here in the model with more predictors. The predictors that tested as having the strongest relationship with the response (that is, they had sufficiently small p-values) were: *zn, dis, rad, black, and medv.*

```r
crime.all <- lm(crim~., data = Boston)
summary(crime.all)
```

```
##
```

```
## 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
```

c) There's quite a lot of difference between coefficients from the single-predictor linear models and those from the fuller model. Some of the coefficients vary little (like *zn*), while other vary a lot more (like *ptratio*). But, standing out is the difference between coefficients for *nox*. In the model where it is the only predictor, the coefficient is 31.25, while in the fuller model the corresponding coefficient is -10.31. It appears that the coefficients related to variables that tested as significant showed the least variance between the two models. That's more evidence that those variables do, in fact, have a significant relationship with the response.

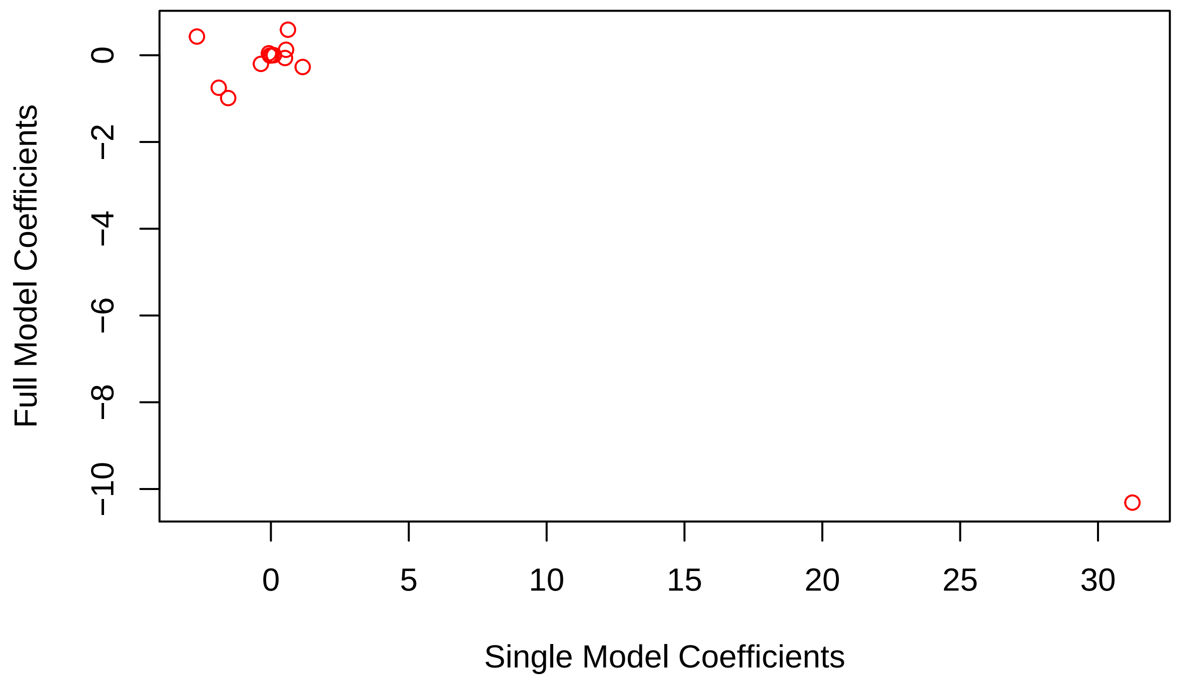
```r
betas <- c(coefficients(B1)[2], coefficients(B2)[2], coefficients(B3)[2], coefficients(B4)[2],
           coefficients(B5)[2], coefficients(B6)[2], coefficients(B7)[2], coefficients(B8)[2],
           coefficients(B9)[2], coefficients(B10)[2], coefficients(B11)[2], coefficients(B12)[2],
           coefficients(B13)[2])

other.betas <- c(coefficients(crime.all)[2:14])

betas
```

```
##          zn       indus        chas         nox          rm         age
## -0.07393498  0.50977633 -1.89277655 31.24853120 -2.68405122  0.10778623
##         dis         rad         tax     ptratio       black       lstat
## -1.55090168  0.61791093  0.02974225  1.15198279 -0.03627964  0.54880478
##        medv
## -0.36315992
```

```
other.betas
```

```
##            zn         indus          chas           nox            rm 
##   0.044855215  -0.063854824  -0.749133611 -10.313534912   0.430130506 
##           age           dis           rad           tax       ptratio 
##   0.001451643  -0.987175726   0.588208591  -0.003780016  -0.271080558 
##         black         lstat          medv 
##  -0.007537505   0.126211376  -0.198886821
```

```
plot(betas, other.betas, xlab = 'Single Model Coefficients', ylab = 'Full Model Coefficients', col = 'r
```


Full Model Coefficients
Single Model Coefficients