Homework 2

Lee Jason

2023-02-03

Libraries

```
library(ISLR2)
library(ggplot2)
```

2.

Chapter 3 Conceptual Exercise 3

(a)

Eqn: $salary = 50 + 20(GPA) + 0.07(IQ) + 35(Level) + 0.01(GPA * IQ) - 10(GPA * Level)$

For fixed GPA and IQ, the equation reduces to: $C + 50 + (35 * Level) - (10G * Level)$ where C and G are constants. Therefore, a high school graduate makes $C + 50$ while a college graduate makes $C + 85 - (10G)$.

$C + 50 = C + 85 - (10 * G)$ finds equivalence when $G = 3.5$. With fixed GPA and IQ, we would expect a high school graduate and college graduate to have the same starting salary when both their GPAs are 3.5.

i.

False. When GPA is below 3.5, college graduates are predicted to have a higher starting salary on average.

ii.

False. When GPA is above 3.5, high school graduates are predicted to have a higher starting salary on average.

iii.

True. As long as GPA is high enough (>3.5), high school graduates are predicted to make a higher staring salary on average.

iv.

False. Our model suggests that college graduates would need a *lower* GPA (<3.5) in order to be predicted to make a higher starting salary on average.

(b)

```
start.sal <- 50+(20*4)+(0.07*110)+(35*1)+(0.01*4*110)-(10*4*1)
paste("Predicted Starting Salary:",start.sal*1000, "dollars")
```

```
## [1] "Predicted Starting Salary: 137100 dollars"
```

(c)

False. A low coefficient does not necessarily imply that there is no significant interaction effect between predictors. A significance test using a t-statistic or some other test statistic to determine a p value would provide better support for an argument for or against an interaction effect. In this case, a "low" coefficient can be caused by differing scales of the predictors. GPA is limited to a value between 1 and 4 while IQ can be measured into the hundreds.

Chapter 3 Conceptual Exercise 4

(a)

Since the true relationship is linear, I don't think there would be huge difference in the RSS for linear vs cubic models since the less flexible model (linear) would still find a good fit with the data. However, in general, the cubic model should be have a smaller training RSS due to the fact that it is more flexible and can accommodate for any noise in the data set, especially because there are relatively few data points to train on. At worst, we would see a RSS very similar to that of the linear model when the data is completely linear.

(b)

I would expect the linear model to perform better (have a lower test RSS) since the cubic model will tend to overfit any training data it is trained on due to higher flexibility and small sample size. The cubic model will have a harder time generalizing to a test set with data it has not seen before.

(c)

It depends. The training RSS for the cubic model would either be better (lower), or at worse very similar to the linear model in this case. Either the relationship tends towards non-linear in which case the cubic model with higher flexibility will produce a closer fit to the data and have a lower training RSS, or the true relationship tends towards perfectly linear in which case the cubic training RSS will approach equivalence to the training RSS of the linear model. Unless the data is perfectly linear, the cubic model is expected to have a lower training RSS due to it's higher flexibility and tendency to overfit. At perfectly linear, they are expected to be equivalent.

(d)

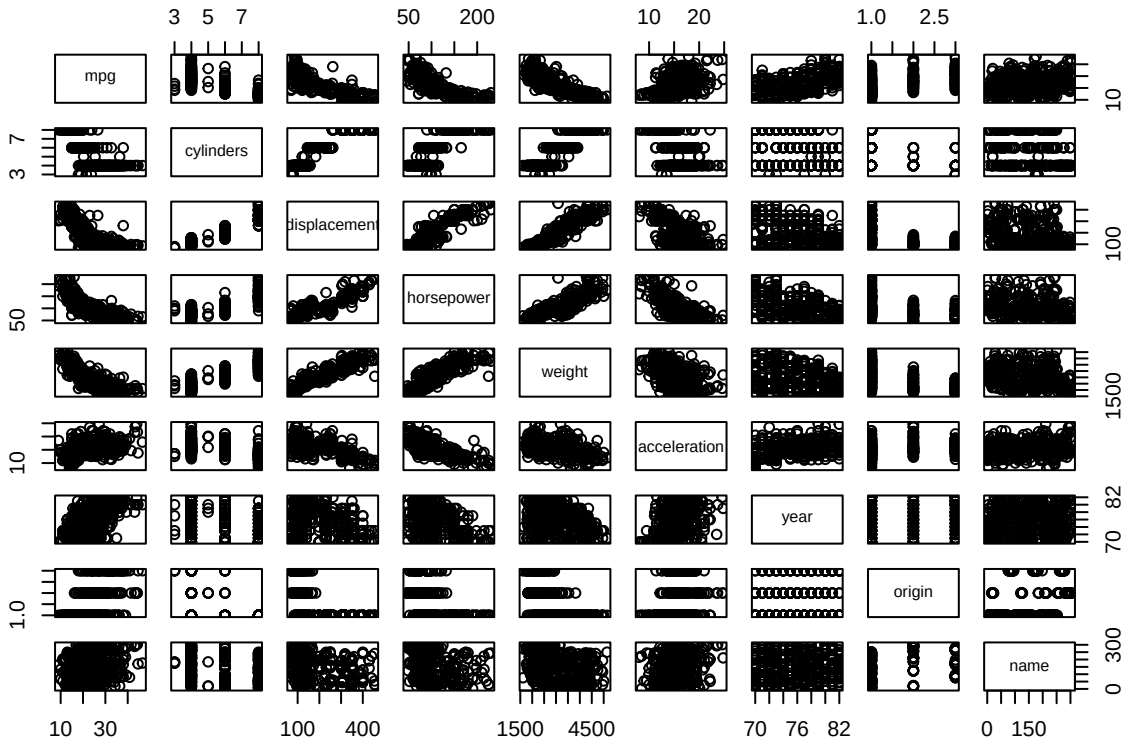
It depends. If the true relationship tends towards linear, we have the same scenario as (b) in which the test RSS of the linear model should be lower as overfitting on a small sample set lowers the ability of a cubic model to predict unseen data while the linear model can approximate well. However, as the true relationship tends towards non-linear, we would expect the test RSS of the cubic model to be lower than the linear model since it closer fits the true relationship compared to linear models despite be more likely to overfit. This trade-off is generally why we see the "U" shape in the Test MSE vs Flexibility.

3.

Chapter 3 Applied Exercise 9

(a)

```
#Scatterplot of all vars from ISLR::Auto dataset
pairs(Auto)
```



(b)

```
#Remove name + get correlation matrix
auto.remove.name <- subset(Auto, select=-c(name))
cor(auto.remove.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
```

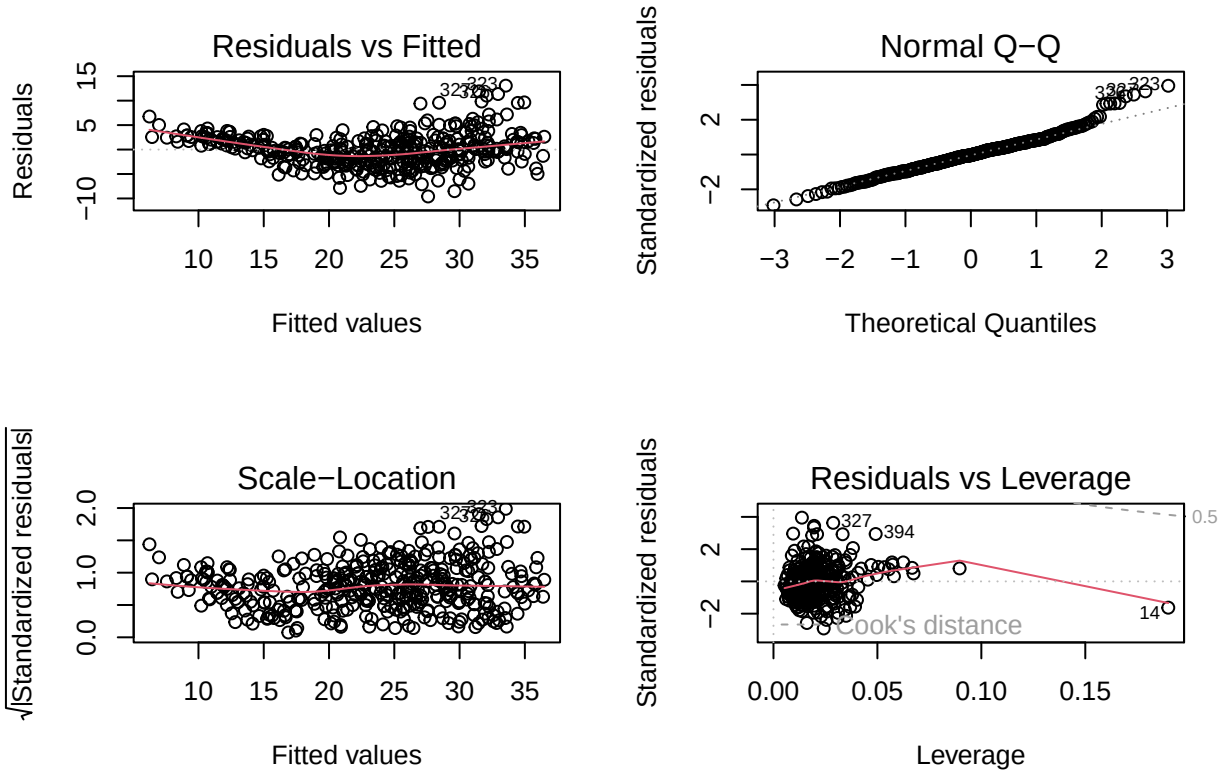
(c)

```
#Linear model w/o name predictor
auto.model <- lm(data=auto.remove.name,mpg~.)
```

- (i): The model summary has a p-value ~0, so we can conclude that there is a significant relationship between the predictors and response. (*Reject* Null Hypothesis $H_0 = \beta_i \; for \; i = (1, 2, ..., p) = 0$)
- (ii): Any predictors with p-values below standard alpha = 0.05 can be considered significant. These include: displacement, weight, year, and origin)
- (iii): The coefficient for year (0.750773) suggests that for every additional year, there is a predicted increase of 0.75 in the response, or mpg

(d)

```
#Fit multiple diagnostic models
par(mfrow=c(2,2))
plot(auto.model)
```



According to the Residuals vs Fitted plot, there seems to a non-linear relationship between the two measurements. This suggests that the model doesn't fully represent some of the relationship between predictors and response due to its inflexibility as a linear model. This unexplained variability would therefore be included as ϵ in the true model. The residual plot doesn't suggest that there are any data points with extreme residuals, suggesting that there aren't any outliers. The leverage plot shows the presence of data points with high leverage with "14" having an extremely high leverage.

(e)

```
interaction.model <- lm(data=auto.remove.name, mpg~.*.)
```

- Individual predictors with significant p values below 0.05 include: displacement, acceleration, origin
- Interaction effects with significant p values below 0.05 include: displacement:year, acceleration:year, acceleration:origin

(f)

```
log.model <- lm(data=auto.remove.name, mpg~.+log(displacement))
sqrt.model <- lm(data=auto.remove.name, mpg~.+sqrt(year))
sq.model <- lm(data=auto.remove.name, mpg~.+I(weight^2))
```

I decided to try transformations on variables that were determined to be significant in the model in (c). By transforming displacement with a log transform, the model does better than the model from (c) with

a greater R^2 value and lower RSE. However, the coefficient estimates drastically changed with most of the predictors having < 0 coefficient estimates. By transforming year with a square root transform, we see a similar trend with a lower RSE and higher R^2 compared to (c). The coefficient estimates were once again all < 0. By transforming weight with a polynomial transform, the same trend overall trend occurs, but we see the best improvement over the model in (c) with an even lower RSE than the log model and an even higher R^2. In general it seems like performing a transformation on a significant predictor increases R^2, reduces RSE, and changes the estimated predictor coefficients to < 0.

Chapter 3 Applied Exercise 10

(a)

```
carseats.model <- lm(data=Carseats,Sales~Price+Urban+US)
```

(b)

```
summary(carseats.model)
```

```
##
## 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
```

- Price: For every additional unit of currency increase in Price assuming fixed US and Urban, there is a corresponding predicted decrease of ~0.054 in the number of unit Sales (54 fewer units sold).
- UrbanYes: If a store is in an Urban location assuming fixed Price and US, there is an expected corresponding decrease of ~0.022 in predicted unit Sales (22 fewer units sold). If the store isn't in an Urban location, there is no expected change in unit Sales.
- USYes: If a store is based in the US assuming fixed Price and Urban, there is an expected corresponding increase of ~1.201 in predicted unit Sales (1200 additional units sold). If the store isn't in the US, there is no expected change in unit Sales.

(c)

Eqn Form: $Sales = 13.043 - 0.054 * Price - 0.022 * Urban + 1.201 * US$ where: - Urban is 1 if a store is in a urban location, else 0 - US is 1 if a store is based in the US, else 0

(d)

The coefficients of the "Price" and "USYes" predictors have with a p-value below the standard 95% confidence level (0.05 alpha). Therefore, we can reject the null hypothesis that the coefficients for these two predictors are zero and conclude there exists a significant relationship between the predictors Price and USYes and the Sales response.

(e)

```
carseats.model2 <- lm(data=Carseats, Sales~Price+US)
```

(f)

Both (a) and (e) fit about the same with R^2 values around 0.239. This suggests a weak, positive fit of the predictors against the response for both. Therefore, the addition or absence of the Urban predictor doesn't seem to improve or worsen our models' ability to predict the response.

(g)

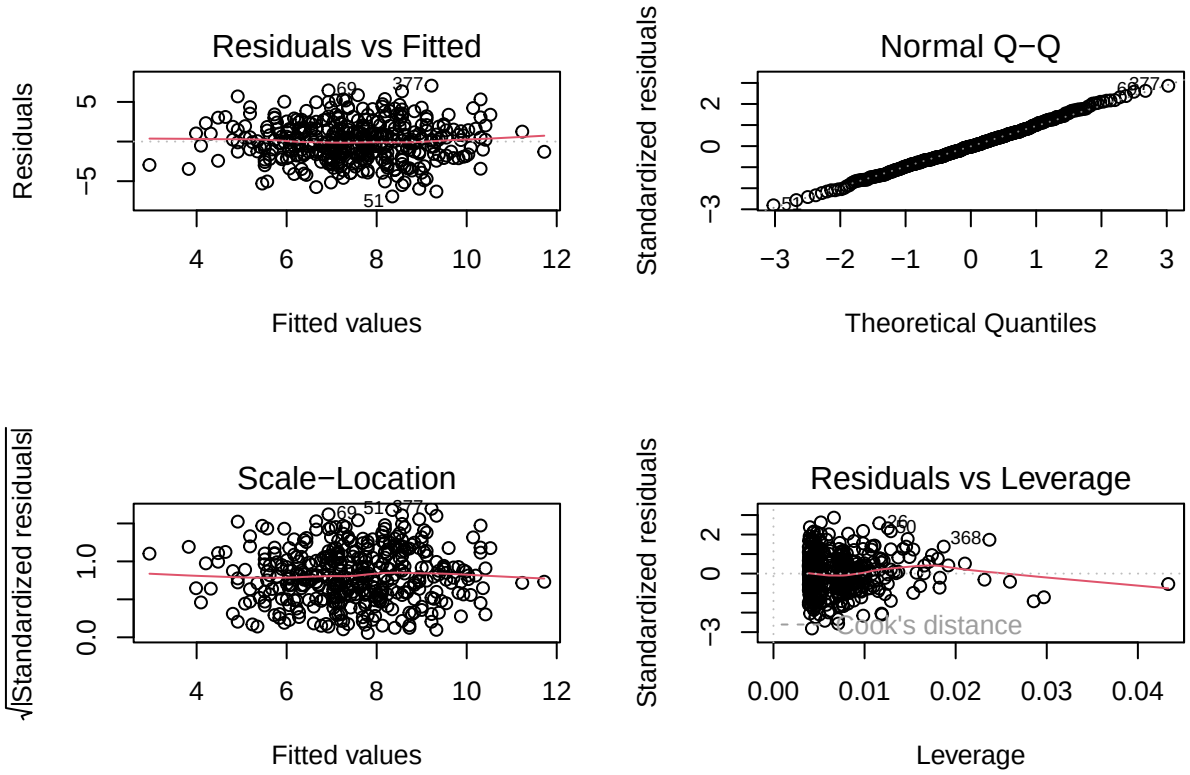
```
confint(carseats.model2)
```

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

Price coefficient has a CI of (-0.065, -0.044). USYes coefficient has a CI of (0.692, 1.708).

(h)

```
par(mfrow=c(2,2))
plot(carseats.model2)
```



In the Residual-Fitted plot, there doesn't seem to be the presence of data points with extremely high residuals, suggesting that there aren't any outliers. The Residuals-Leverage plot does show the presence of data points with high leverage.

Chapter 3 Applied Exercise 13

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

(a)

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

(b)

```
my.sd <- sqrt(0.25) #Standard Deviation is the sqrt of variance
eps <- rnorm(100, sd=my.sd)
```

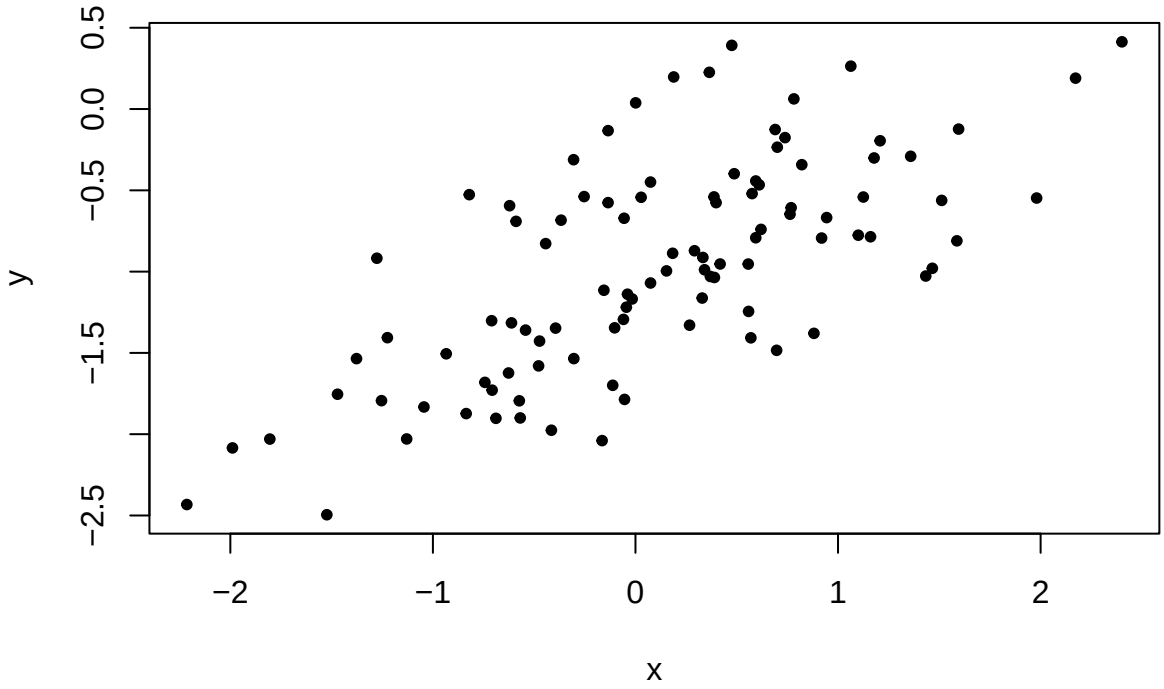
(c)

```
y <- -1 + (0.5*x) + eps
y.len <- length(y)
cat("Y length:",y.len,"\nBeta-0: -1\nBeta-1: 0.5")
```

```
## Y length: 100
## Beta-0: -1
## Beta-1: 0.5
```

(d)

```
plot(x, y, pch=20)
```



- There is an overall positive linear relationship between x and y.
- The scales of x and y are about the same
- The data isn't perfectly linear, there is some noise in the data set

(e)

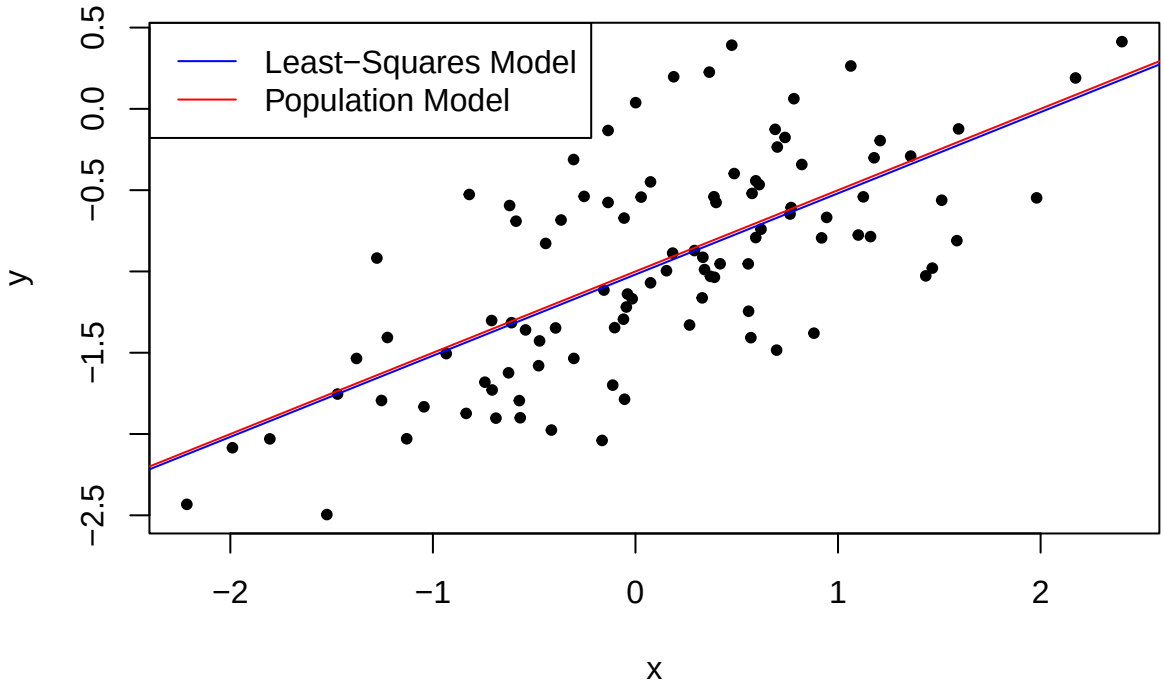
```
ls.model <- lm(y~x)
```

- $\hat{\beta}_0 = -1.019$
- $\hat{\beta} = 0.4995$

In our least-squares model, our $\hat{\beta}_0$ is nearly the same as our population model β_0 as it is about -1. Our least squares $\hat{\beta}_1$ is also pretty much equal to our population β_1 as it is about 0.5.

(f)

```
plot(x, y, pch=20)
abline(ls.model, col = "blue")
abline(a=-1,b=0.5, col = "red")
legend("topleft", legend = c("Least-Squares Model", "Population Model"), col=c("blue","red"), lwd=c(1,1
```



(g)

```
my.poly.model <- lm(y~x+I(x^2))
```

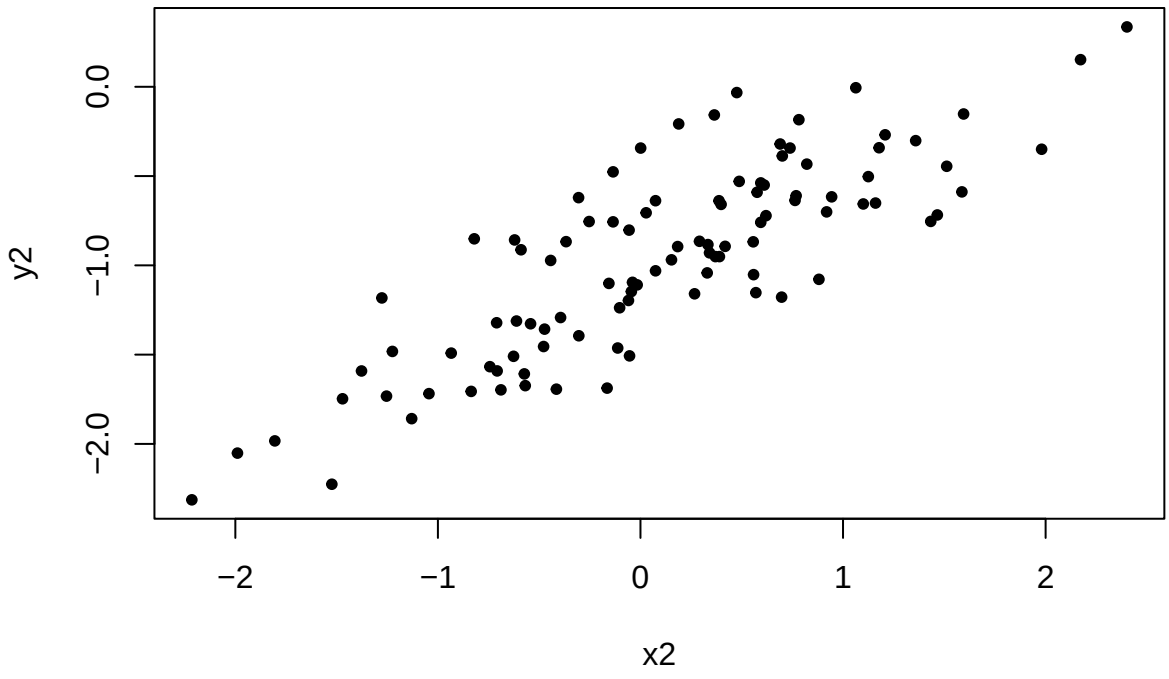
No. A summary of the model including the polynomial term indicates that the polynomial term does not improve the fit as it has a p-value greater than 0.05.

(h)

```
set.seed(1)
#Repeat a-c
x2 <- rnorm(100)
eps2 <- rnorm(100, sd=sqrt(0.1)) #Reduce variance to 0.10
y2 <- -1 + (0.5*x2) + eps2
y2.len <- length(y2)
cat("Y2 length:",y2.len,"\nBeta-0: -1\nBeta-1: 0.5")
```

```
## Y2 length: 100
## Beta-0: -1
## Beta-1: 0.5
```

```
#Repeat d
plot(x2, y2, pch=20)
```



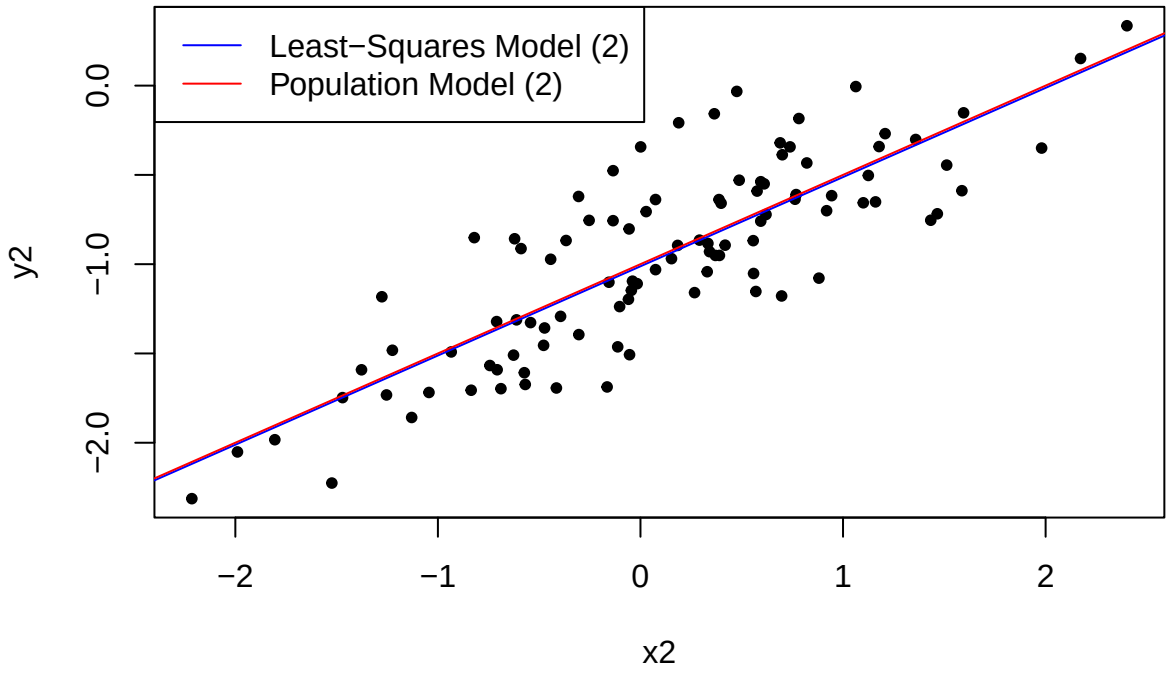
- The plot is nearly the same as that from (d), but there is a noticeable decrease in noise in the data as the points are "tighter" in the middle.

```
#Repeat e
ls.model2 <- lm(y2~x2)
```

- $\hat{\beta}_0 = -1.012$
- $\hat{\beta} = 0.4997$

In our least-squares model, our $\hat{\beta}_0$ is nearly the same as our population model β_0 as it is about -1. Our least squares $\hat{\beta}_1$ is also pretty much equal to our population β_1 as it is about 0.5. Though small, with smaller variance the estimates for our coefficient and intercept were closer to the true values compared to our initial variance.

```
#Repeat f
plot(x2, y2, pch=20)
abline(ls.model2, col = "blue")
abline(a=-1,b=0.5, col = "red")
legend("topleft", legend = c("Least-Squares Model (2)", "Population Model (2)"), col=c("blue","red"), lw
```



- We can see that the two regression lines are a bit closer compared to the one in (f).

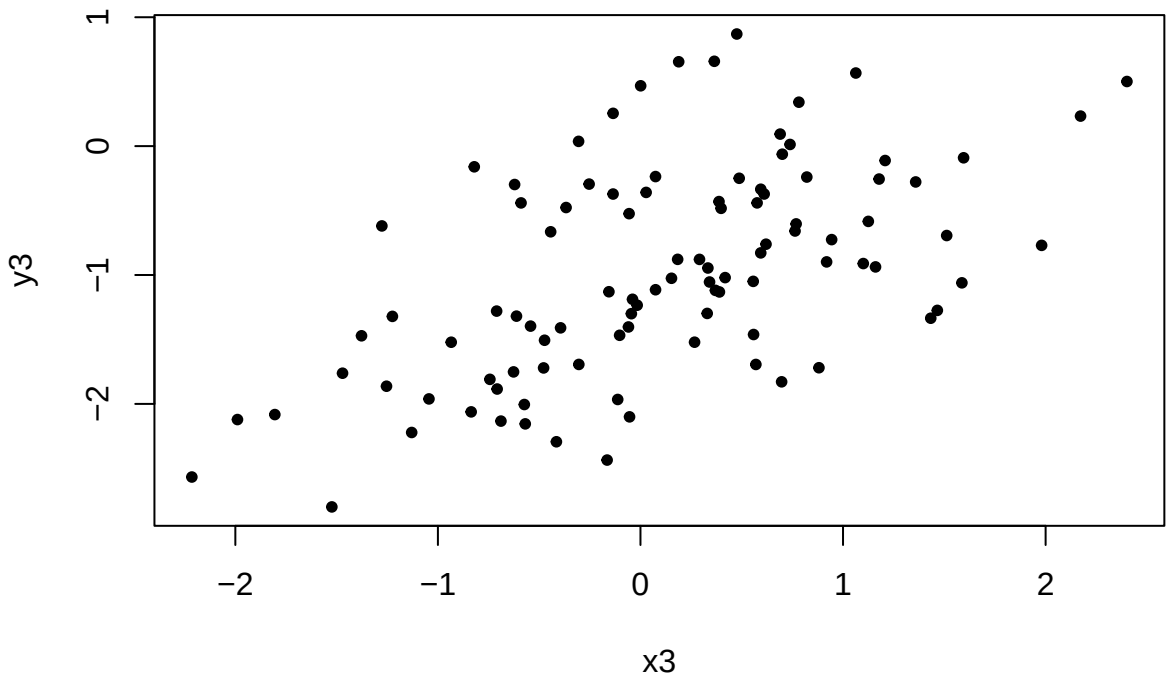
(i)

```
set.seed(1)
#Repeat a-c
x3 <- rnorm(100)
eps3 <- rnorm(100, sd=sqrt(0.5)) #Increase variance to 0.5
y3 <- -1 + (0.5*x3) + eps3
```

```r
y3.len <- length(y3)
cat("Y length:",y3.len,"\nBeta-0: -1\nBeta-1: 0.5")
```

```
## Y length: 100
## Beta-0: -1
## Beta-1: 0.5
```

```r
#Repeat d
plot(x3, y3, pch=20)
```



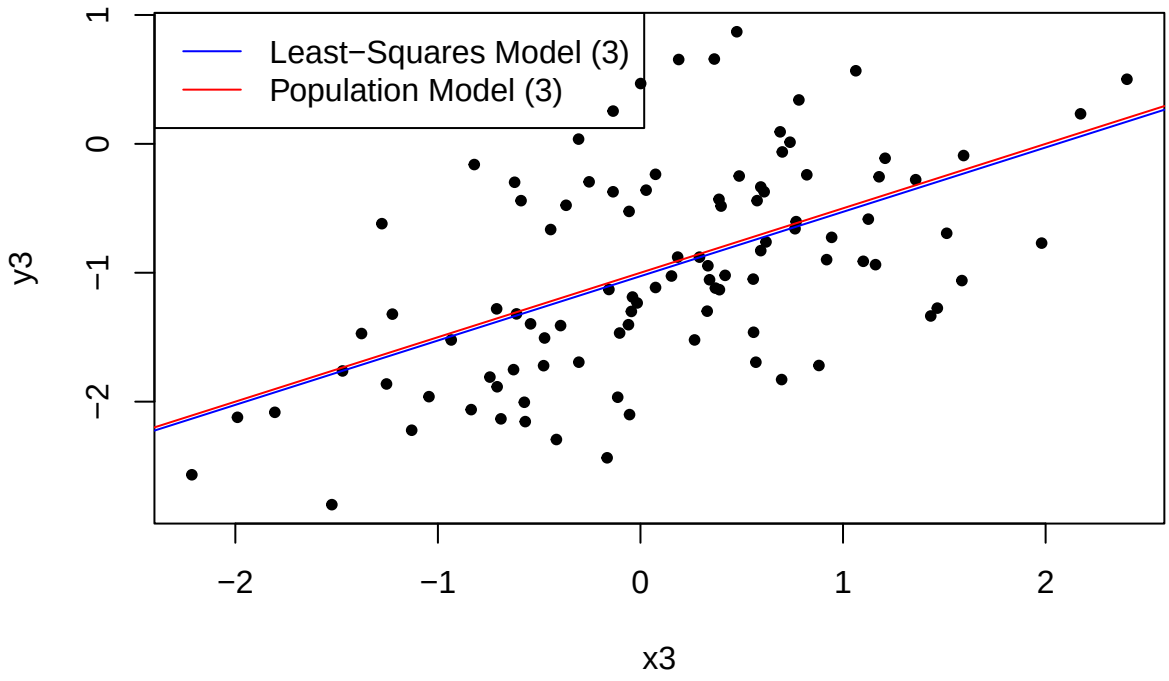
- The plot is nearly the same as that from (d) and (h), but there is a noticeable increase in noise in the data as the points are much more sporadic in the middle.

```r
ls.model3 <- lm(y3~x3)
```

- $\hat{\beta}_0 = -1.027$
- $\hat{\beta} = 0.4993$

In our least-squares model, our $\hat{\beta}_0$ is nearly the same as our population model β_0 as it is about -1. Our least squares $\hat{\beta}_1$ is also pretty much equal to our population β_1 as it is about 0.5. Though small, with larger variance the estimates for our coefficient and intercept were further from the true values compared to our initial variance.

```
plot(x3, y3, pch=20)
abline(ls.model3, col = "blue")
abline(a=-1,b=0.5, col = "red")
legend("topleft", legend = c("Least-Squares Model (3)", "Population Model (3)"), col=c("blue","red"), lw
```



- We can see that the two regression lines are a bit further apart compared to the one in (f).

(j)

```
confint(ls.model) #Initial Model
```

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

```
confint(ls.model2) #Less Variance
```

```
##                  2.5 %     97.5 %
## (Intercept) -1.0727832 -0.9510557
## x2           0.4320613  0.5672681
```

```
confint(ls.model3) #More Variance
```

```
##                   2.5 %     97.5 %
## (Intercept) -1.1627482 -0.8905572
## x3           0.3480843  0.6504160
```

Compared to the initial Model, the model with less variance had a larger CI for the intercept, but a tighter CI for the x predictor. The model with higher variance compared to the initial Model saw the opposite with a tighter CI for the intercept, and a larger CI for the x predictor.

Chapter 3 Applied Exercise 14

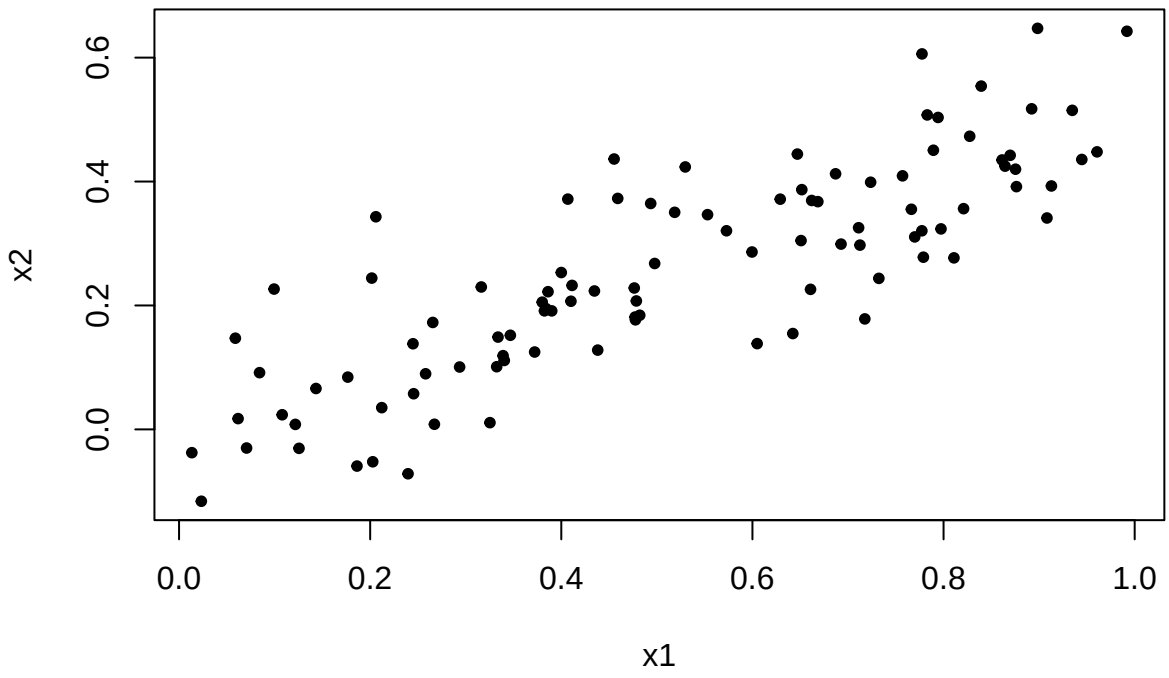
(a)

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

General Form: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$ where: - $\beta 0 = 2$ - $\beta 1 = 2$ - $\beta 2 = 0.3$

(b)

```
plot(x1,x2,pch=20)
```




```
paste("Correlation between x1 and x2: ", cor(x1,x2))
```

```
## [1] "Correlation between x1 and x2:  0.835121242463113"
```

(c)

```
ls.model2 <- lm(y~x1+x2)
```

- $\hat{\beta}_0 = 2.1305$
- $\hat{\beta}_1 = 1.4396$
- $\hat{\beta}_2 = 1.0097$

While the intercept $\hat{\beta}_0$ is around the same as the true intercept, the coefficients of $\hat{\beta}_1$ and $\hat{\beta}_2$ were not close to their true counterparts. At a standard 95% confidence level (0.05 alpha), only x1 is below alpha, so we can reject $\beta_1 = 0$, but we fail to reject $\beta_2 = 0$.

(d)

```
ls.x1Model <- lm(y~x1)
```

- $\hat{\beta}_0 = 2.1124$

- $\hat{\beta}_1 = 1.9759$

$\hat{\beta}_0$ is around the same as the true β_0 while $\hat{\beta}_1$ is also close to the true β_1. At an alpha of 0.05, we can reject $H_0 : \beta_1 = 0$ since the p-value is below alpha.

(e)

```
ls.x2Model <- lm(y~x2)
```

- $\hat{\beta}_0 = 2.3899$
- $\hat{\beta}_1 = 2.8996$

$\hat{\beta}_0$ is somewhat close to the true β_0 while $\hat{\beta}_1$ is not close to the true β_1. At an alpha of 0.05, we can still reject $H_0 : \beta_1 = 0$ since the p-value is below alpha.

(f)

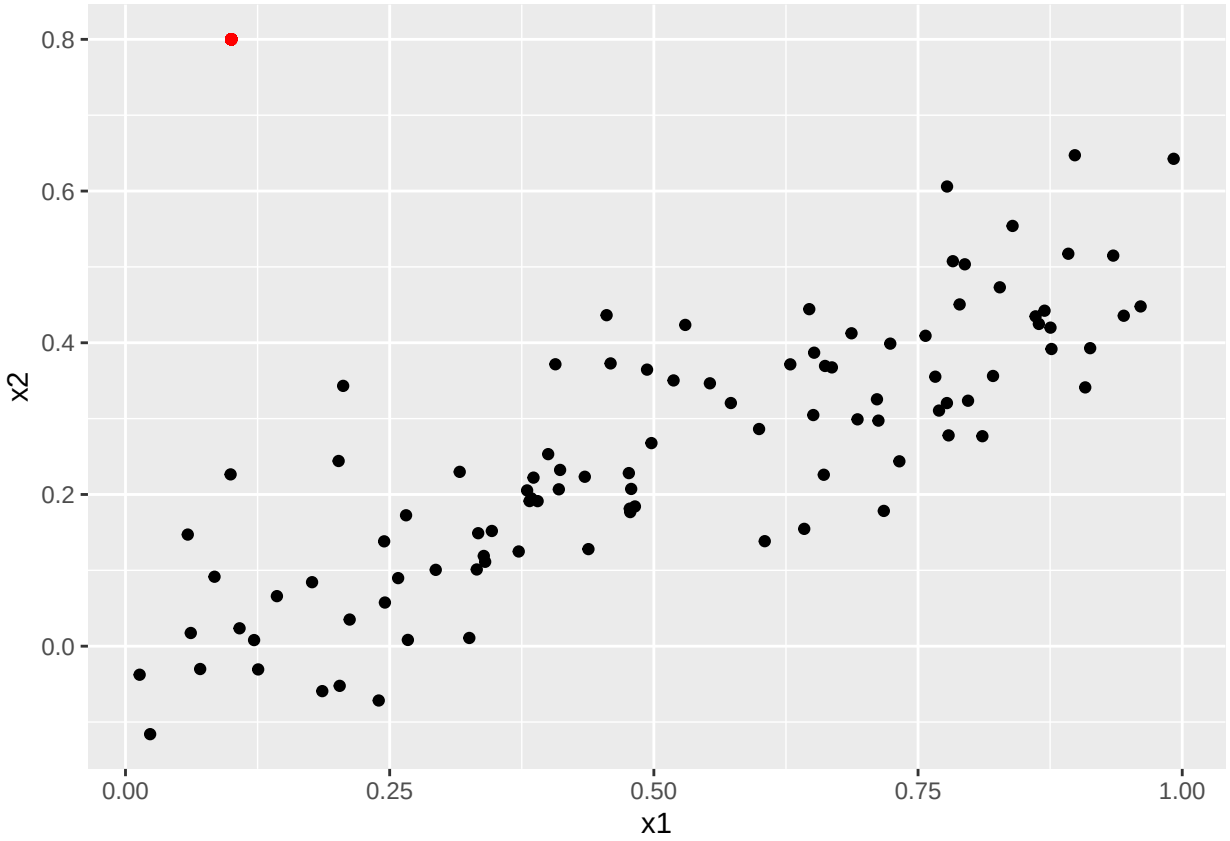
There is some differences in the models as we observed that with both predictors, x2 is not significant. However, by itself it is significant. The results don't necessarily contradict each other as our models just signify that when both are present, x1 is able to explain enough of the response variability that x2 doesn't have a significant impact in explaining the remaining response variability. Without x1, x2 is able to explain enough of the response variability to be significant.

(g)

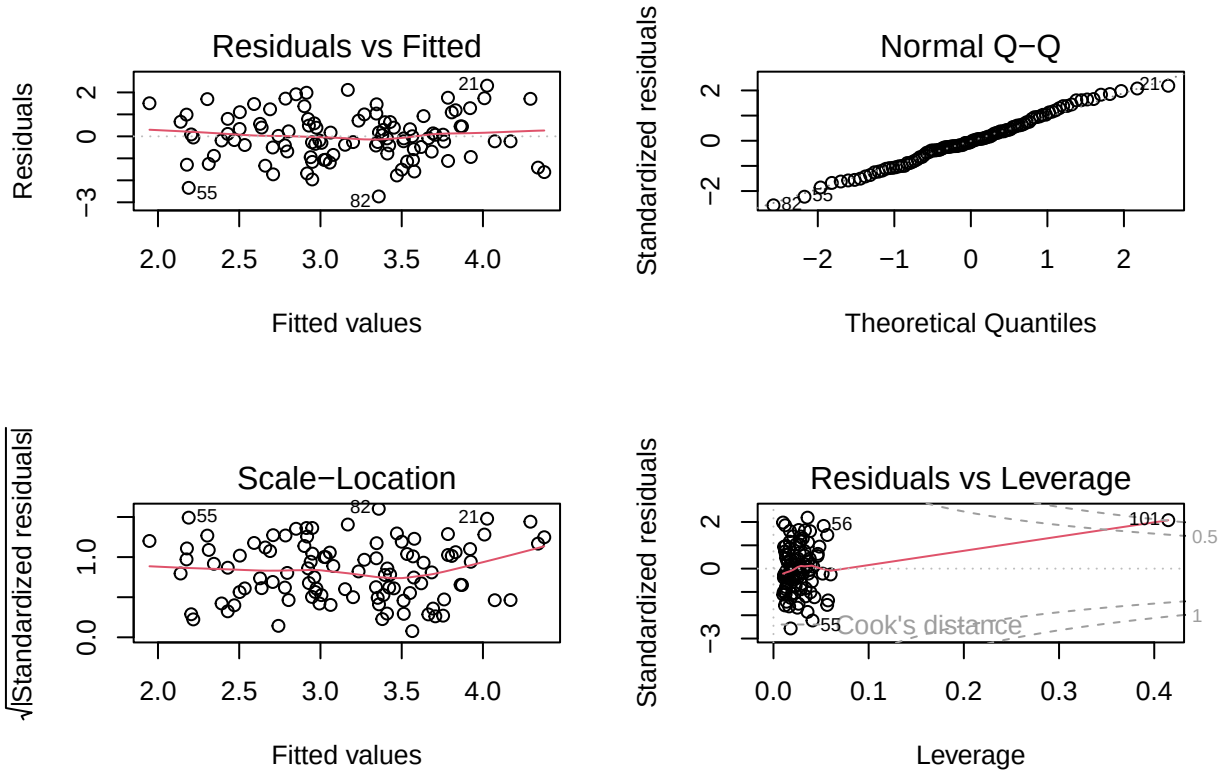
```
#From problem description
x1 <- c(x1, 0.1)
x2 <- c(x2, 0.8)
y <- c(y, 6)

ls.model3 <- lm(y~x1+x2)
ls.x1Model2 <- lm(y~x1)
ls.x2Model2 <- lm(y~x2)
```

In the model with both predictors, we see that x2 becomes the only significant predictor ($p < 0.05$) which is the opposite of what we observed in (c). In each model with individual predictors, the predictors are still significant which matches what we observed in (d) and (e). However, in the models with individual predictors with the new data point, x2 showed a better fit compared to the model in (e) with a higher R^2 while x1 had a lower R^2 than the corresponding model in (d).

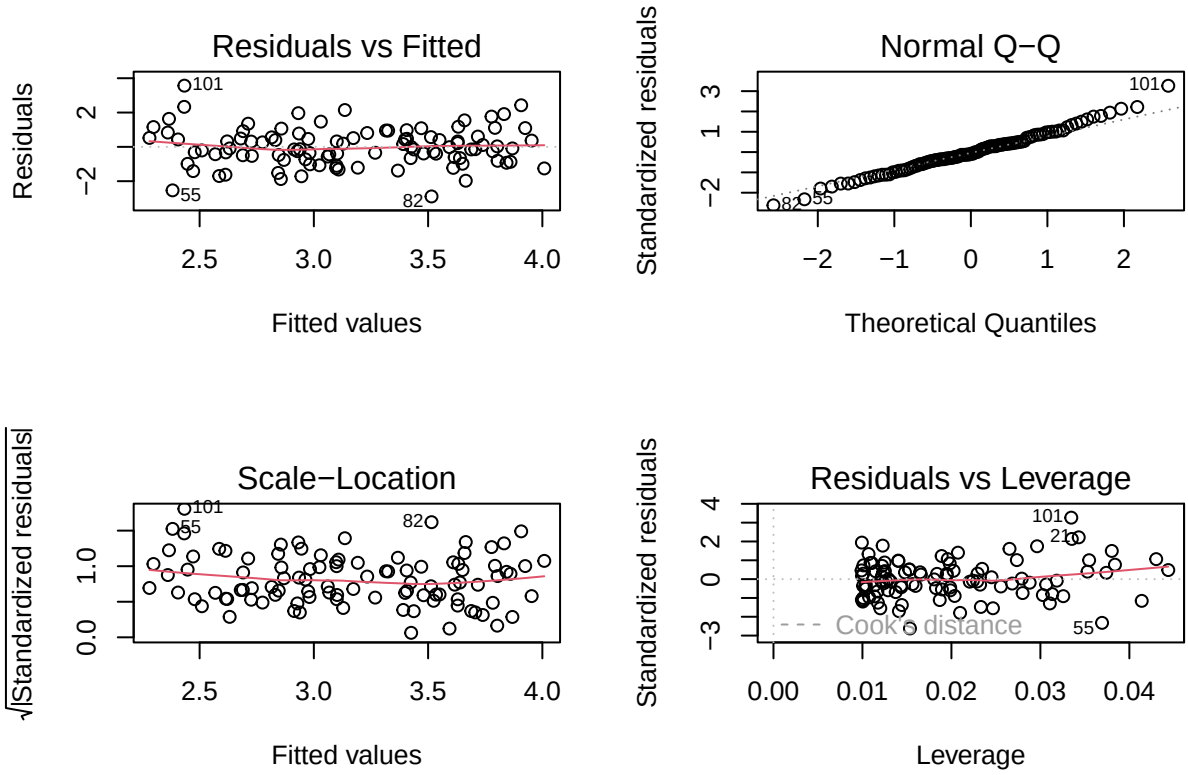
```
ggplot(data=data.frame(x1,x2), mapping = aes(x1,x2))+
  geom_point() +
  geom_point(aes(x=0.1,y=0.8), color = "red")
```



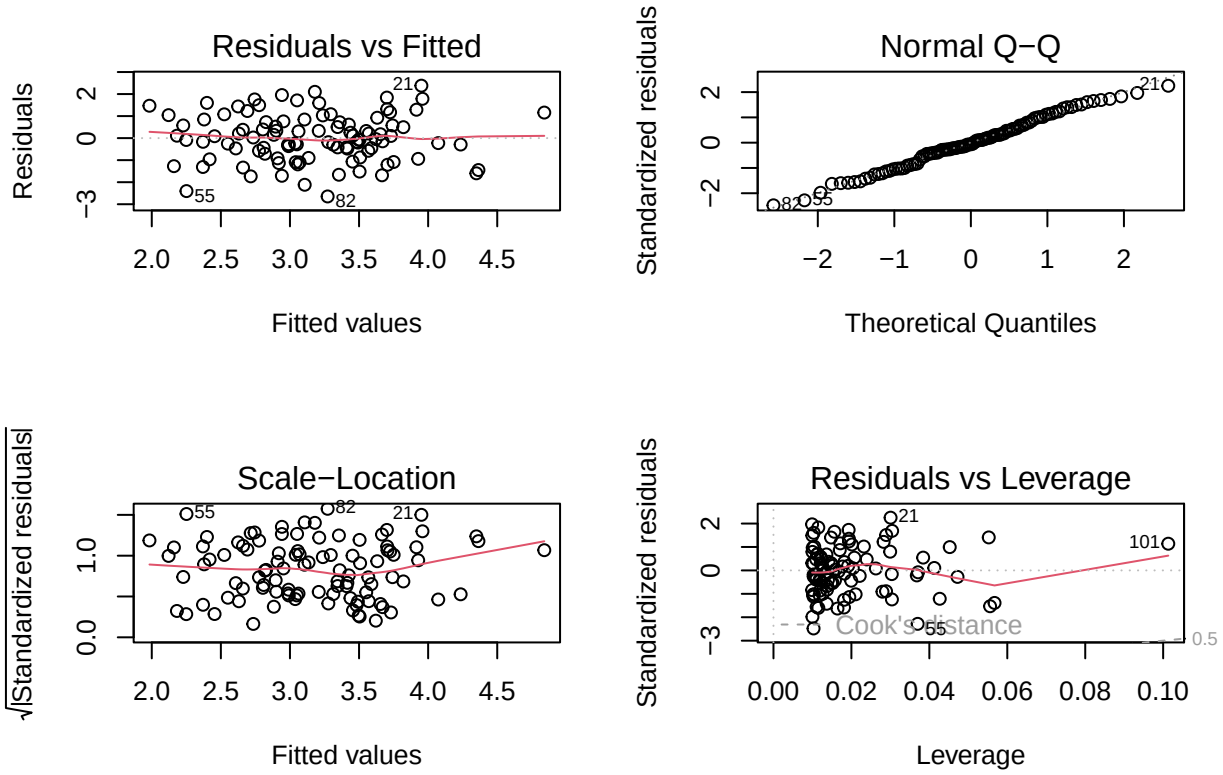

```
par(mfrow=c(2,2))
plot(ls.model3)
```




```
par(mfrow=c(2,2))
plot(ls.x1Model2)
```




```
par(mfrow=c(2,2))
plot(ls.x2Model2)
```



The new value is definitely an outlier for all three models as seen from our x1 vs x2 plot. However, in our new models, we can see that it is only a high leverage point in the model with both x1 and x2 and the model with just x2 as there is a point (101) in both that have a leverage much larger than any other point. There is no distinguishable point with high leverage in the model with just x1.

4.

In Lecture 2 (covering the material from ISLR Chapter 2) we discussed the fact that more flexible models always fit better on the training data, regardless of how "good" a model they may actually be (i.e. regardless of how well they would fit on new data, not used to build the model). This exercise is designed to emphasize that point in the context of linear models.

Function for 4a and 4b

```
create.norm.df <- function(size = 25){

  #initialize a default data.frame object to fill in
  create.norm.df <- as.data.frame(matrix(data=0,nrow=size,ncol=size))

  #Fill the data.frame by adding rows of 25 #s from the normal distribution
  for(i in 1:size){
    create.norm.df[i,] <- rnorm(size)
  }
```

```
  #Set Response Column:
  # 1)Selecting random column (by sampling a col #)
  # 2)Appending the col column-wise to the end of the data.frame
  # 3)Remove the original column selected to be response
  rand_response <- sample(1:size, 1)
  create.norm.df <- cbind(create.norm.df, create.norm.df[,rand_response])
  create.norm.df <- create.norm.df[,-rand_response]

  #Set column names
  colnames(create.norm.df) <- c(1:24, "y")
  return (create.norm.df)
}
```

(a)

Generate 25 variables, each of which consists of 25 random samples from a standard normal. Store these variables in a data frame – call it df.train – and randomly select one variable to be the response – rename it y. (The end result should be a data frame with 25 observations on 25 variables but with no relationships between any of the variables.)

```
#Set seed + create df.train
set.seed(99)
df.train <- create.norm.df()
```

(b)

Repeat step (a) to create a test set called df.test

```
#Set seed + create df.test
set.seed(100)
df.test <- create.norm.df()
```

(c)

Write a loop that will successively linearly regress y on one additional predictor each time through. That is, the first time through the loop you should build a linear model with only one predictor (the first one in your data frame). The ith time through the loop, you should build a linear model where y is regressed on the first i predictors. Record the training and test error each time so that at the end of the procedure you have two vectors (call them MSE.train and MSE.test) that contain the MSEs from each model.

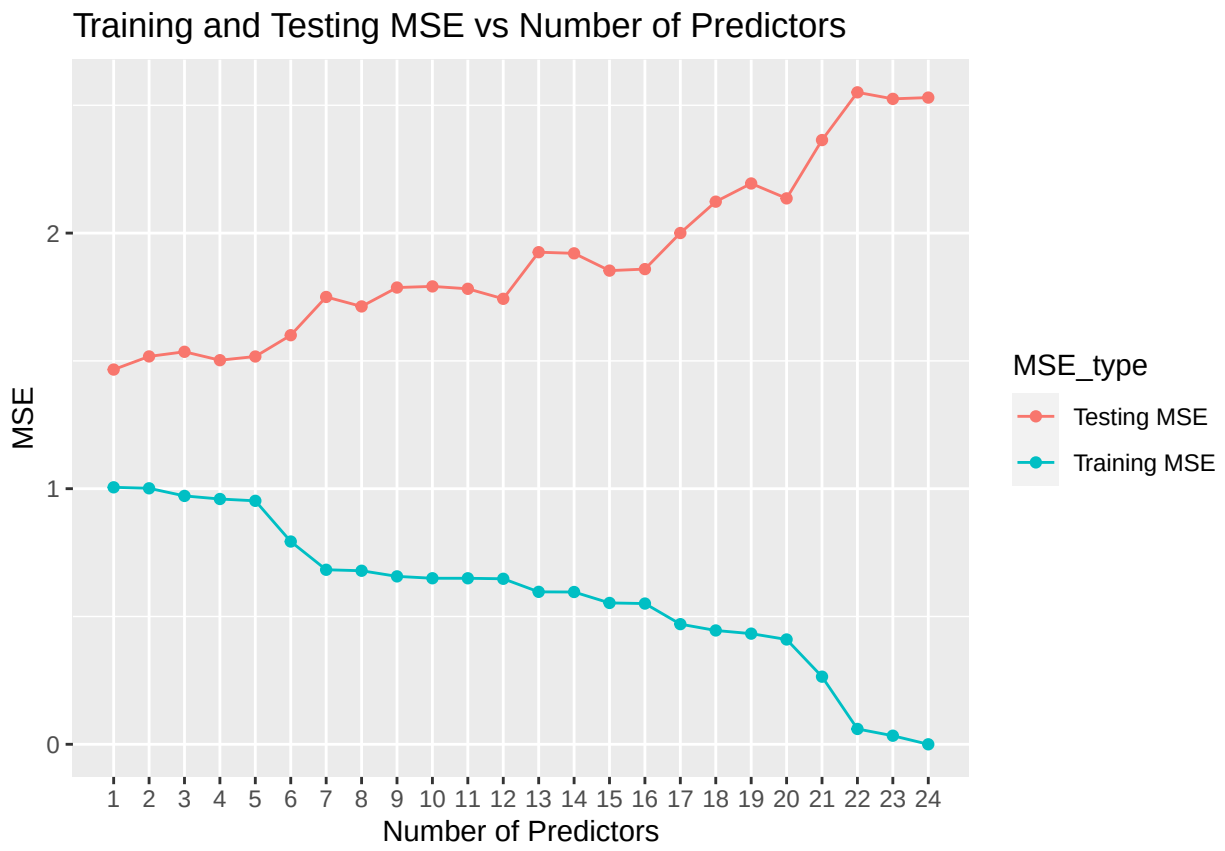
```
#Define vectors to hold MSE values for train and test
MSE.train <- vector(mode="numeric", length=24)
MSE.test <- vector(mode="numeric", length=24)

#For loop to add additional predictor each time through
for(i in 1:24){
  #Formula object of the regression equation for predictors 1 through i
  my.forumula <- as.formula(paste("y~",paste0("`",colnames(df.train)[c(1:i)],
                                              collapse="+","`")))

  #Build the linear model
```

```
  my.model <- lm(data=df.train, my.forumula)
  my.intercept <- coef(my.model)[1]
  my.coefs <- as.matrix(coef(my.model)[-1])

  #Calculate the MSE for the training data set + add to appropriate vector
  train.predictions <- predict(my.model, data=df.train[,c(1:i)])
  train.residuals.sq <- (df.train["y"]-train.predictions)^2
  MSE.train[i] <- sum(train.residuals.sq) / 25

  #Calculate the MSE for the testing data set + add to appropriate vector
  test.predictions <- predict(my.model, data=df.test[,c(1:i)])
  test.residuals.sq <- (df.test["y"]-test.predictions)^2
  MSE.test[i] <- sum(test.residuals.sq) / 25
}
```

(d)

Plot the training and test errors vs the linear model size (number of predictors) on the same plot in different colors. Add a legend to the plot to distinguish them.

```
#"Tidy" the data to plot
desig <- c(rep("Training MSE",24), rep("Testing MSE",24))
MSE.df <- data.frame(MSE_type=desig, num.pred=rep(1:24,2), vals=c(MSE.train, MSE.test))

#Plot using ggplot2 library
ggplot(data=MSE.df, mapping = aes(x=num.pred, y=vals, color = MSE_type)) +
  geom_point() + geom_line() +
  scale_x_continuous(breaks = 1:24, minor_breaks = NULL) +
  labs(title="Training and Testing MSE vs Number of Predictors", x="Number of Predictors", y="MSE")
```



(e)

What happens to the training error as more predictors are added to the model? What about the test error?

As the number of predictors increases, the training error steadily decreases with relatively steep decreases from 4 to 5 and 20 to 22 predictors. On the other hand, test error gradually increases as more predictors are added to the model.