Stat 435: HW 1

Pat McCornack

2022-09-19

Question 1

1a

First we must load the *Auto* dataset from the *ISLR* library.

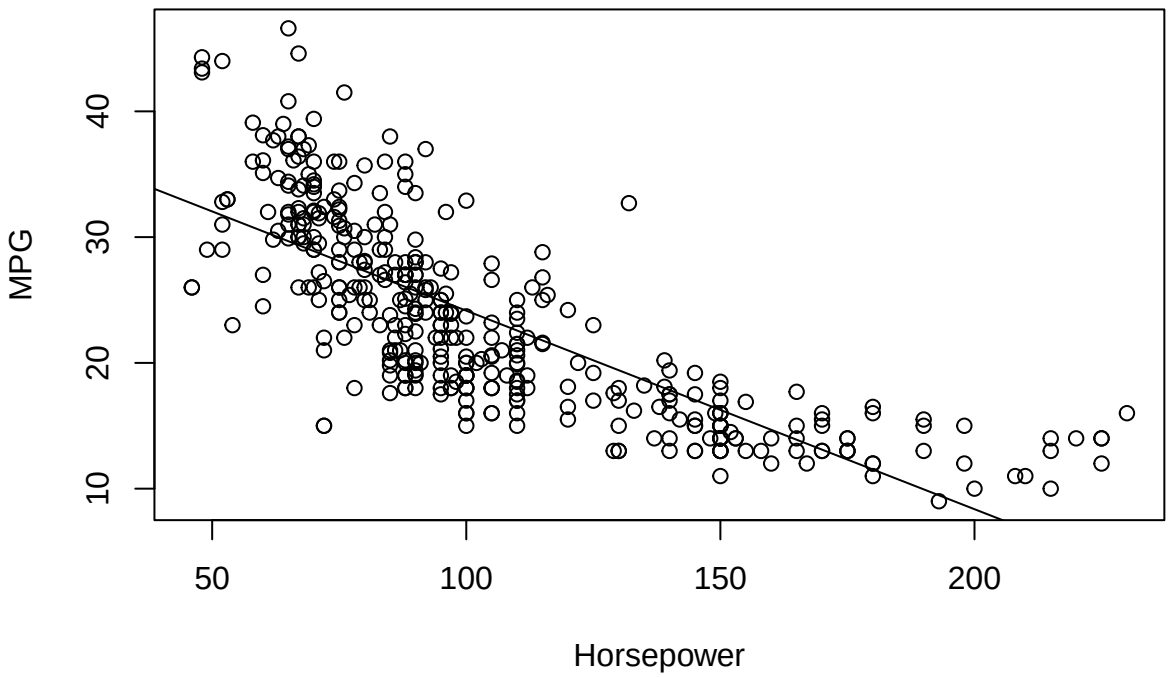
```
library(ISLR)
data("Auto")
```

Then we create the linear regression and get summary statistics as well as plot the data in order to visualize it.

```
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
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

```
plot(Auto$horsepower, Auto$mpg, xlab = "Horsepower", ylab = "MPG")
abline(lm.fit)
```



From the summary statistics we see:

Significance: There is a statistically significant relationship between the predictor and the response value based on the p-value which is much less than 0.05. The interpretation of this is that there's a very small chance (<2e-16) that the given data was observed despite there being no association.

Strength: With a regression coefficient `B1 = -0.15` we see that this linear relationship is not very strong. However, by looking at the plot we can see that a linear model may not be the best fit for this data

Direction: The relationship is negative. A car with more horsepower tends to have a lower mpg rating.

Prediction: Using the coefficients from the model, we can predict that a car with a horsepower of 98 will have mpg of 24.47. The as.numeric function was used to remove the 'horsepower' label from the output to avoid confusion. (Alternatively, I could have used `predict(lm.fit, data.frame(horsepower = 98))` for this prediction.)

```
prediction = as.numeric((coef(lm.fit)[2] * 98 + coef(lm.fit)[1]))
prediction
```

```
## [1] 24.46708
```

Confidence Intervals: The prediction interval associated with a horsepower of 98 is between 14.6708 - 34.12476. This means there is a 95% confidence that a new value of mpg associated with a horsepower of 98 would fall within this range (i.e. If we did an experiment where we tested the mpg of a car with 98 horsepower 100 times then 95 of those values would fall in this range.)

```
predict(lm.fit, data.frame(horsepower = 98), interval ="prediction")
```

```
##        fit     lwr      upr
## 1 24.46708 14.8094 34.12476
```

The confidence interval for the regression coefficient is that there is 95% confidence that it will be between -0.17 and -.14.

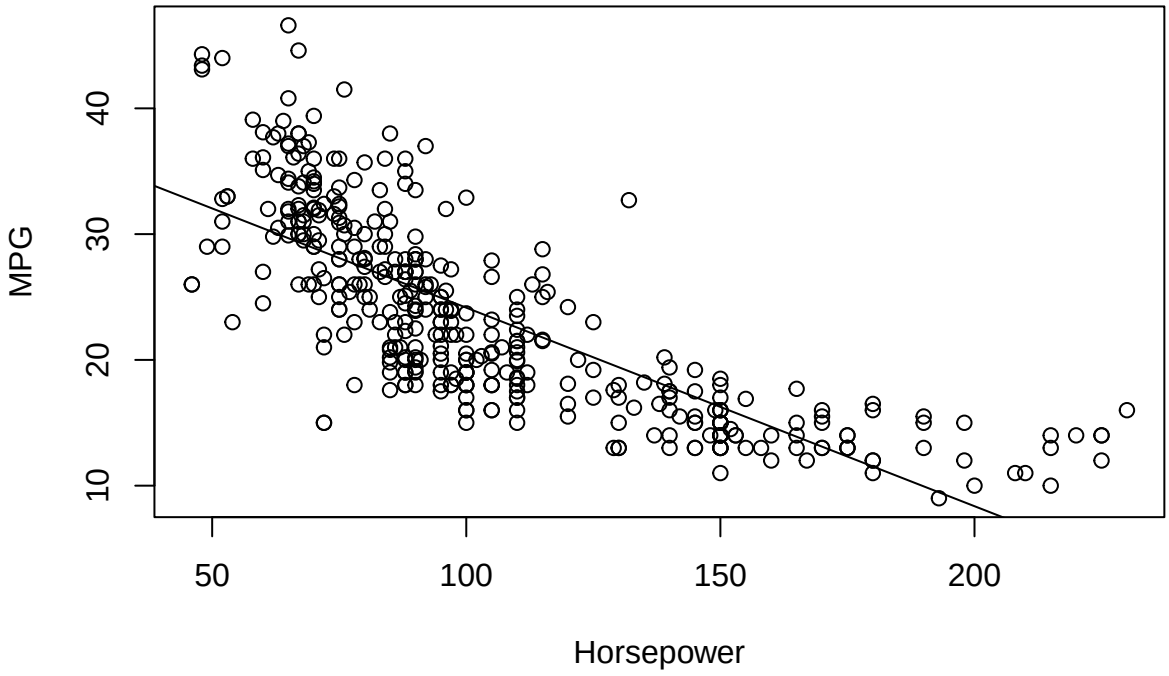
```
confint(lm.fit)
```

```
##                 2.5 %     97.5 %
## (Intercept) 38.525212 41.3465103
## horsepower  -0.170517 -0.1451725
```

1b

As before, the plot with the regression line is as follows:

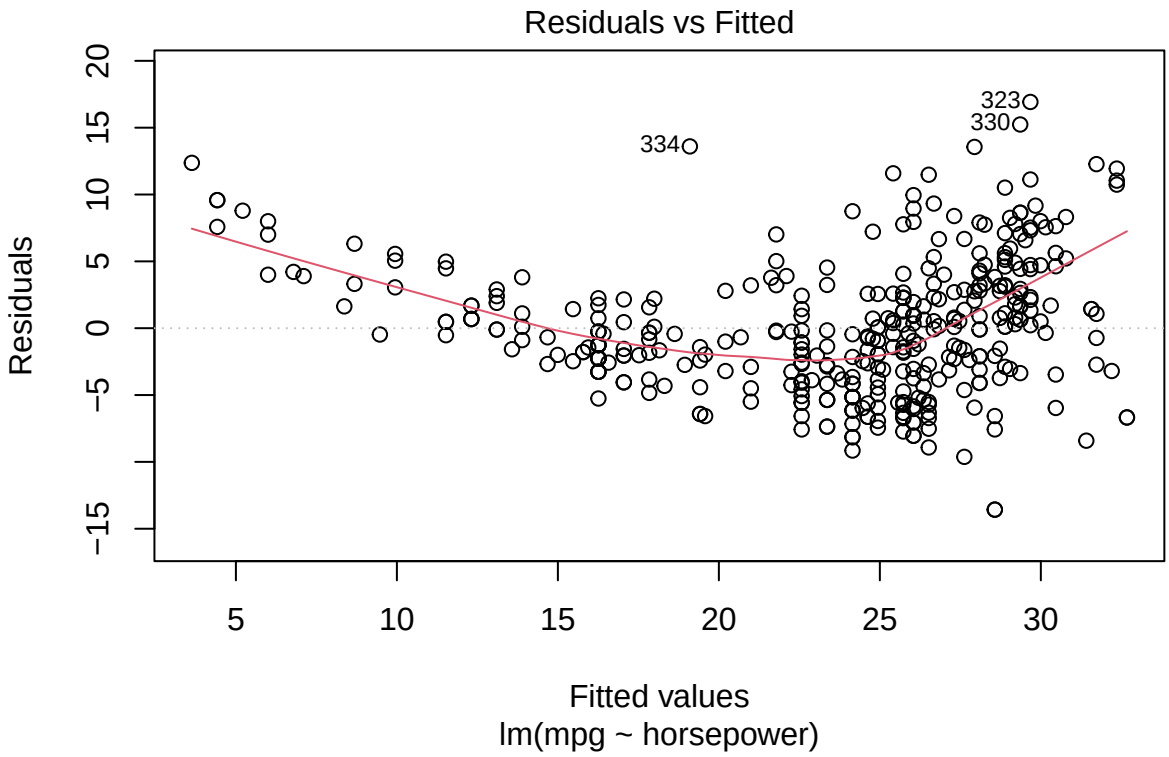
```
plot(Auto$horsepower, Auto$mpg, xlab = "Horsepower", ylab = "MPG")
abline(lm.fit)
```

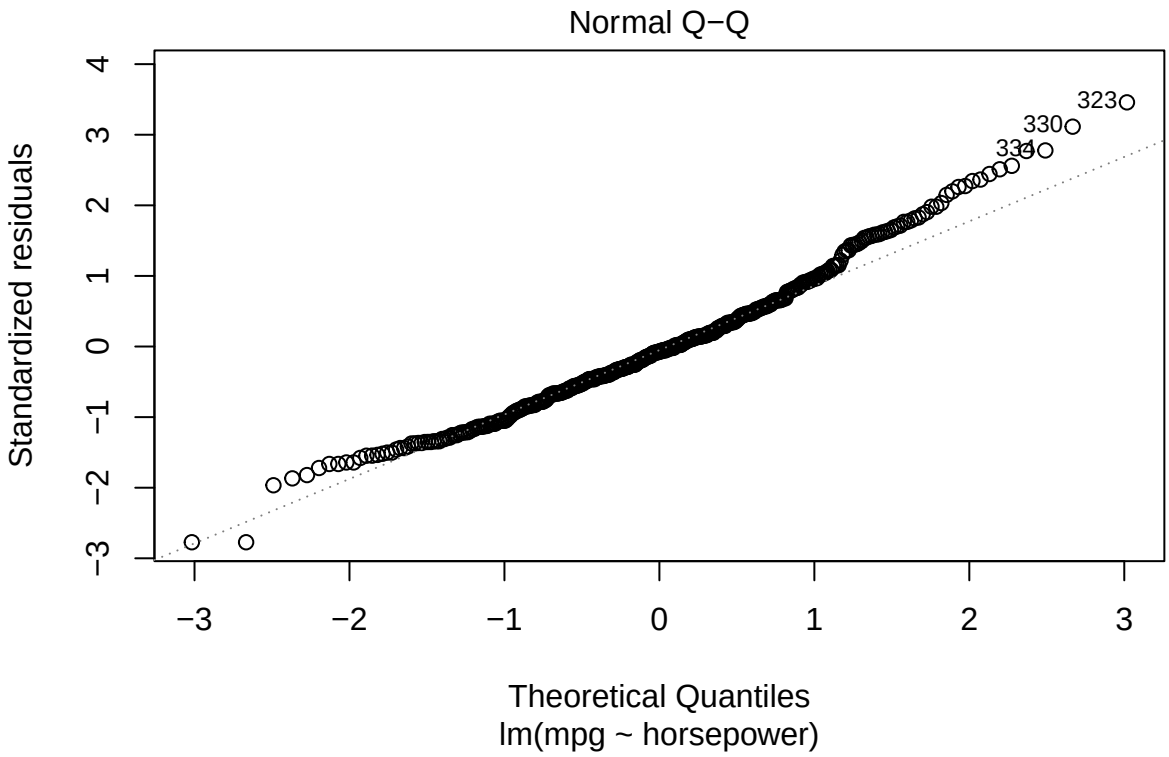


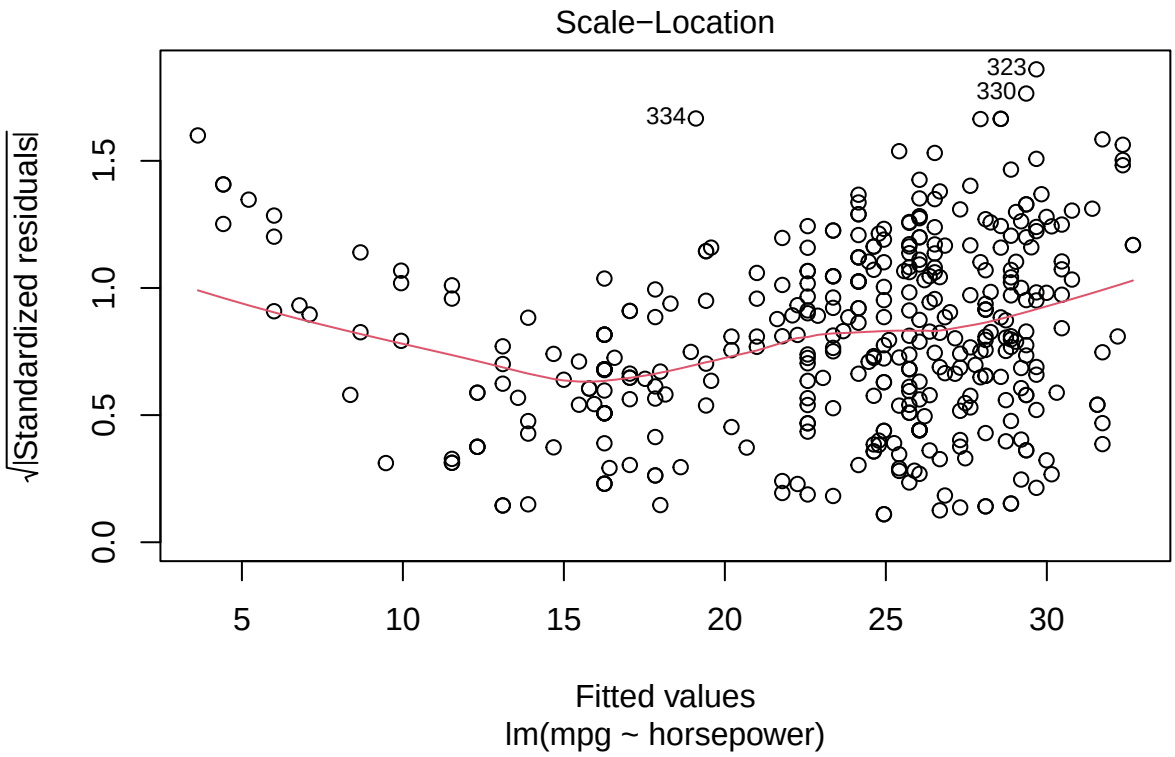
1c

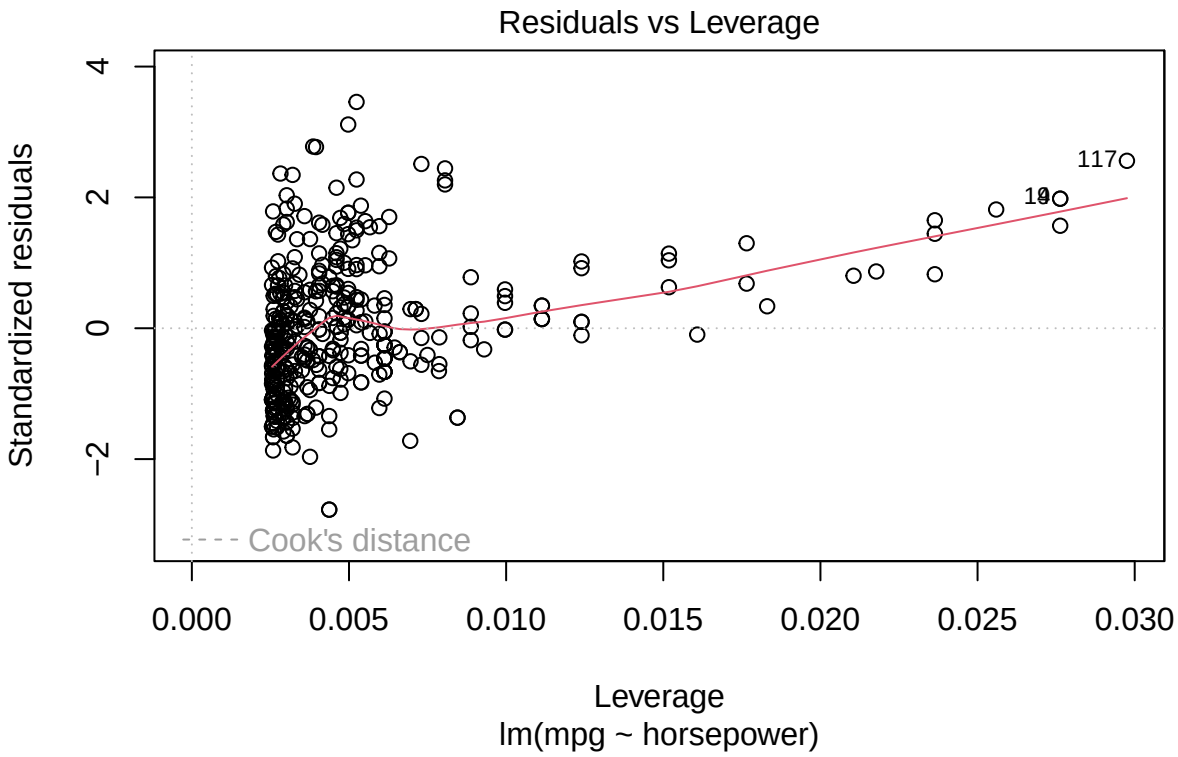
The following creates a plot of the fitted values against the residuals. We can clearly that there is a trend in the data using the "Residuals vs Fitted" plot and therefore **we can assume that the model is not valid** because the underlying assumption of linearity was violated. If there was no trend then the residuals would be spread around 0.

```
plot(lm.fit)
```


Residuals vs Fitted
Residuals
Fitted values
lm(mpg ~ horsepower)
334
323
330
−15
−5
0
5
10
15
20
5
10
15
20
25
30


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(mpg ~ horsepower)
323
330
334
−3
−2
−1
0
1
2
3
4


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
5
10
15
20
25
30
323
330
334
Fitted values
lm(mpg ~ horsepower)



Question 2

2a

First we use the standard normal distribution to generate a vector x of 100 values centered around 0 with a standard deviation of 1.

```
set.seed(1)  # Ensure the same "random numbers" generated each time
x = rnorm(100, 0, 1)
```

2b

Then we use a normal distribution centered around 0 with a standard deviation of .25 to create a vector of 100 observations which will act as the noise term eps:

```
eps = rnorm(100, 0, 0.25)
```

2c

These object can then be used to generate a vector Y based on the model `Y = -1 + 0.5X + eps`. The code following then finds the length of the vector y (which is 100 based on the lengths of X and eps).

```
y = -1 + 0.5*x + eps
length(y)
```

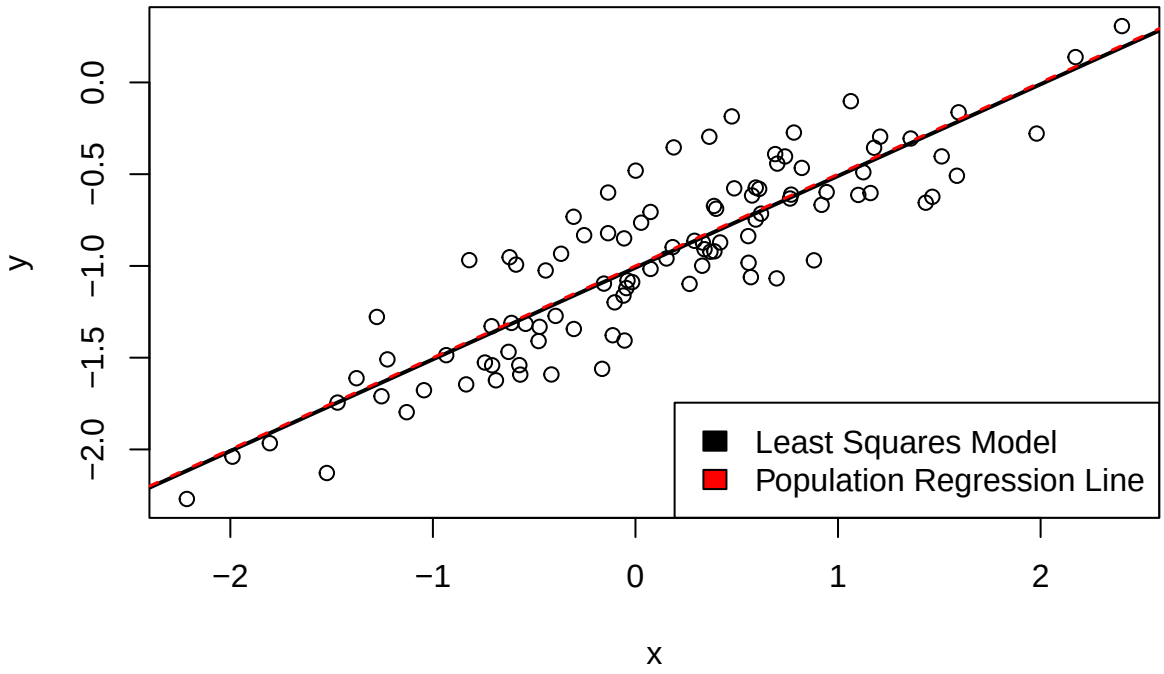
```
## [1] 100
```

In this model $\beta_0 = -1$ and $\beta_1 = 0.5$.

2d

We can then use `lm()` to create a least squares model and fit it to a plot of the x, y values. We also use the known values of β_0 and β_1 to fit the population regression line.

```
y = -1 + 0.5*x + eps
lm.sim_fit = lm(y ~ x)  # Least squares model
plot(x, y)
abline(lm.sim_fit, lwd = 2)  # Fit least squares model
abline(-1, 0.5, lwd =1.5, lty = 2, col="Red")  # Fit the population regression line
legend(x="bottomright",
       legend = c("Least Squares Model", "Population Regression Line"),
       fill = c("black", "red"))
```



2e

In considering the RSS for the quadratic model vs the linear model we would expect them to be about the same. This is because the least squares model attempts to minimze the RSS. However, if we were to look a plot of the residuals vs. the fitted values there would be a clear trend in the quadratic model while there would be no trend in the linear model.

```
x2 = x^2
quad_model <- lm(y ~ x + x2)
summary(quad_model)
```

```
##
## Call:
## lm(formula = y ~ x + x2)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4913 -0.1563 -0.0322  0.1451  0.5675
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.98582    0.02941 -33.516   <2e-16 ***
## x            0.50429    0.02700  18.680   <2e-16 ***
## x2          -0.02973    0.02119  -1.403    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2395 on 97 degrees of freedom
## Multiple R-squared:  0.7828, Adjusted R-squared:  0.7784
## F-statistic: 174.8 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
summary(lm.sim_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
```

2f

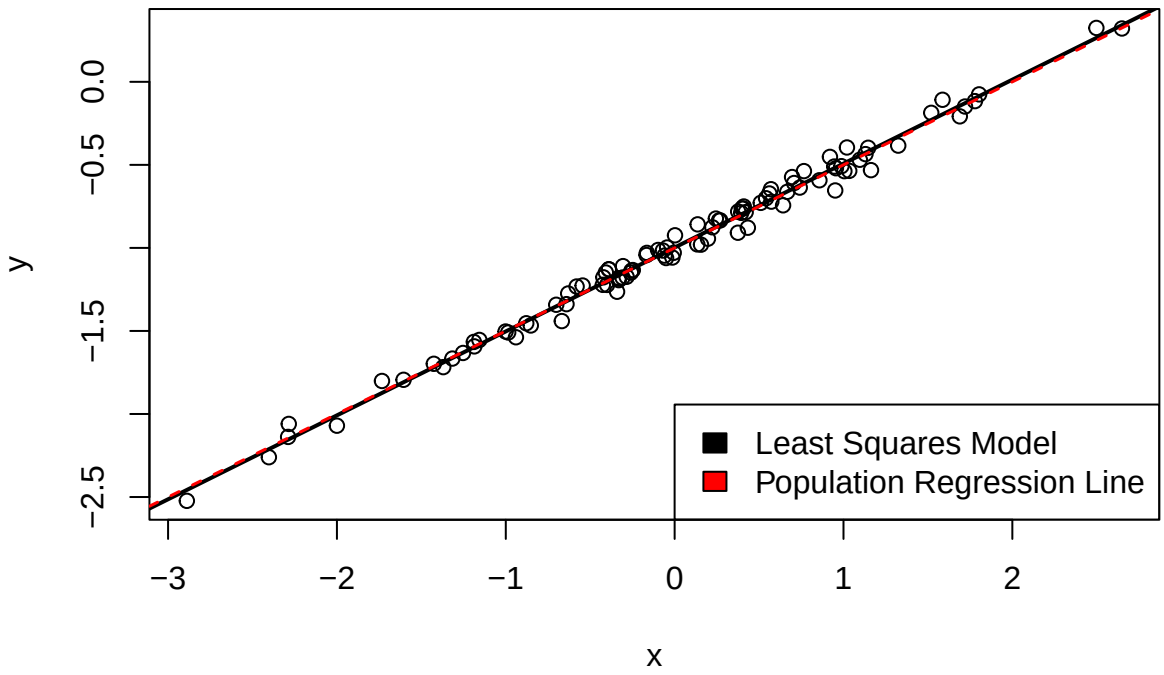
Using the F statistic we would expect both models to still yield significant results, but the statistic for the multiple linear regression would be around half.

2g

When the noise in the model is decreased by decreasing the variance of the noise vector normal distribution, we see that the fit of both models goes up. In particular, the regression coefficient for x becomes even closer to be nearly the same value and the R^2 value for each becomes the same very high value. In this model over 99% of the variation in y can be explained by the variation in x, which makes sense given the variance of the noise vector was set to .01.

```
# Linear model
x = rnorm(100, 0, 1)
eps = rnorm(100, 0, 0.05)
y = -1 + 0.5*x + eps
```

```r
lm.sim_fit = lm(y ~ x)  # Least squares model
plot(x, y)
abline(lm.sim_fit, lwd = 2)  # Fit least squares model
abline(-1, 0.5, lwd =1.5, lty = 2, col="Red")  # Fit the population regression line
legend(x="bottomright",
       legend = c("Least Squares Model", "Population Regression Line"),
       fill = c("black", "red"))
```




```r
# Quadratic Model
x2 = x^2
quad_model <- lm(y ~ x + x2)
summary(quad_model)
```

```
##
## Call:
## lm(formula = y ~ x + x2)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.137065 -0.027658 -0.001063  0.032886  0.096560
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -0.996391   0.005986 -166.462   <2e-16 ***
## x            0.505096   0.004872  103.678   <2e-16 ***
## x2          -0.001114   0.003120   -0.357    0.722
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04975 on 97 degrees of freedom
## Multiple R-squared:  0.9912, Adjusted R-squared:  0.991
## F-statistic:  5462 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
summary(lm.sim_fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.137090 -0.028070 -0.000874  0.033987  0.092421
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.997578   0.004955  -201.3   <2e-16 ***
## x            0.505311   0.004813   105.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04953 on 98 degrees of freedom
## Multiple R-squared:  0.9912, Adjusted R-squared:  0.9911
## F-statistic: 1.102e+04 on 1 and 98 DF,  p-value: < 2.2e-16
```

Question 3

3a.

Load the Boston data set. For this problem: Response: Per Capita Crime Rate (crim) Predictors: All other variables in data set

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

Fit a simple linear regression model for each predictor

```
attach(Boston)  # Make the Boston dataset available to the following functions:

zn_lm = lm(crim ~ zn)
indus_lm = lm(crim ~ indus)
chas_lm = lm(crim ~ chas)
nox_lm = lm(crim ~ nox)
rm_lm = lm(crim ~ rm)
age_lm = lm(crim ~ age)
dis_lm = lm(crim ~ dis)
rad_lm = lm(crim ~ rad)
tax_lm = lm(crim ~ tax)
ptratio_lm = lm(crim ~ ptratio)
black_lm = lm(crim ~ black)
lstat_lm = lm(crim ~ lstat)
medv_lm = lm(crim ~ medv)
```

```
summary(zn_lm)
```

```
##
## Call:
## lm(formula = crim ~ zn)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.429 -4.222 -2.620  1.250 84.523
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.45369    0.41722  10.675  < 2e-16 ***
## zn          -0.07393    0.01609  -4.594 5.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.435 on 504 degrees of freedom
## Multiple R-squared:  0.04019,    Adjusted R-squared:  0.03828
## F-statistic:  21.1 on 1 and 504 DF,  p-value: 5.506e-06
```

```
summary(indus_lm)
```

```
##
## Call:
## lm(formula = crim ~ indus)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.972  -2.698  -0.736   0.712  81.813
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.06374    0.66723  -3.093  0.00209 **
## indus        0.50978    0.05102   9.991  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.866 on 504 degrees of freedom
## Multiple R-squared:  0.1653, Adjusted R-squared:  0.1637
## F-statistic: 99.82 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(chas_lm)
```

```
##
## Call:
## lm(formula = crim ~ chas)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.738 -3.661 -3.435  0.018 85.232
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.7444     0.3961   9.453   <2e-16 ***
```

```
## chas         -1.8928     1.5061  -1.257    0.209
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.597 on 504 degrees of freedom
## Multiple R-squared:  0.003124,   Adjusted R-squared:  0.001146
## F-statistic: 1.579 on 1 and 504 DF,  p-value: 0.2094
```

```
summary(nox_lm)
```

```
##
## Call:
## lm(formula = crim ~ nox)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.371  -2.738  -0.974   0.559  81.728
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -13.720      1.699  -8.073 5.08e-15 ***
## nox           31.249      2.999  10.419  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.81 on 504 degrees of freedom
## Multiple R-squared:  0.1772, Adjusted R-squared:  0.1756
## F-statistic: 108.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(rm_lm)
```

```
##
## Call:
## lm(formula = crim ~ rm)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.604 -3.952 -2.654  0.989 87.197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   20.482      3.365   6.088 2.27e-09 ***
## rm            -2.684      0.532  -5.045 6.35e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.401 on 504 degrees of freedom
## Multiple R-squared:  0.04807,    Adjusted R-squared:  0.04618
## F-statistic: 25.45 on 1 and 504 DF,  p-value: 6.347e-07
```

```
summary(age_lm)
```

```
##
## Call:
## lm(formula = crim ~ age)
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.789 -4.257 -1.230  1.527 82.849
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.77791    0.94398  -4.002 7.22e-05 ***
## age          0.10779    0.01274   8.463 2.85e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.057 on 504 degrees of freedom
## Multiple R-squared:  0.1244, Adjusted R-squared:  0.1227
## F-statistic: 71.62 on 1 and 504 DF,  p-value: 2.855e-16
```

```
summary(dis_lm)
```

```
##
## Call:
## lm(formula = crim ~ dis)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.708 -4.134 -1.527  1.516 81.674
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.4993     0.7304  13.006   <2e-16 ***
## dis          -1.5509     0.1683  -9.213   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.965 on 504 degrees of freedom
## Multiple R-squared:  0.1441, Adjusted R-squared:  0.1425
## F-statistic: 84.89 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(rad_lm)
```

```
##
## Call:
## lm(formula = crim ~ rad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.164  -1.381  -0.141   0.660  76.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.28716    0.44348  -5.157 3.61e-07 ***
## rad          0.61791    0.03433  17.998  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.718 on 504 degrees of freedom
## Multiple R-squared:  0.3913, Adjusted R-squared:   0.39
```

```
## F-statistic: 323.9 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(tax_lm)
```

```
##
## Call:
## lm(formula = crim ~ tax)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.513  -2.738  -0.194   1.065  77.696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.528369   0.815809  -10.45   <2e-16 ***
## tax          0.029742   0.001847   16.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.997 on 504 degrees of freedom
## Multiple R-squared:  0.3396, Adjusted R-squared:  0.3383
## F-statistic: 259.2 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(ptratio_lm)
```

```
##
## Call:
## lm(formula = crim ~ ptratio)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.654 -3.985 -1.912  1.825 83.353
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.6469     3.1473  -5.607 3.40e-08 ***
## ptratio       1.1520     0.1694   6.801 2.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.24 on 504 degrees of freedom
## Multiple R-squared:  0.08407,    Adjusted R-squared:  0.08225
## F-statistic: 46.26 on 1 and 504 DF,  p-value: 2.943e-11
```

```
summary(black_lm)
```

```
##
## Call:
## lm(formula = crim ~ black)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.756  -2.299  -2.095  -1.296  86.822
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 16.553529   1.425903  11.609   <2e-16 ***
## black       -0.036280   0.003873  -9.367   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.946 on 504 degrees of freedom
## Multiple R-squared:  0.1483, Adjusted R-squared:  0.1466
## F-statistic: 87.74 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(lstat_lm)
```

```
##
## Call:
## lm(formula = crim ~ lstat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.925  -2.822  -0.664   1.079  82.862
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.33054    0.69376  -4.801 2.09e-06 ***
## lstat        0.54880    0.04776  11.491  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.664 on 504 degrees of freedom
## Multiple R-squared:  0.2076, Adjusted R-squared:  0.206
## F-statistic:   132 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(medv_lm)
```

```
##
## Call:
## lm(formula = crim ~ medv)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.071 -4.022 -2.343  1.298 80.957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.79654    0.93419   12.63   <2e-16 ***
## medv        -0.36316    0.03839   -9.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.934 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491
## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16
```

With the exception of the chas (a dummy variable indicating whether a tract is on the Charles River), every model had a statistically significant relationship between predictor and response based on the p-value. The chas model had a p-value of .209, but this is unsurprising given that that dataset is categorical using a dummy variable. While the models were statistically significant they all had low R^2 values. The highest R^2 was .39 for the rad (index of accessibility to radial highways) data while most of the rest were closer to .15.

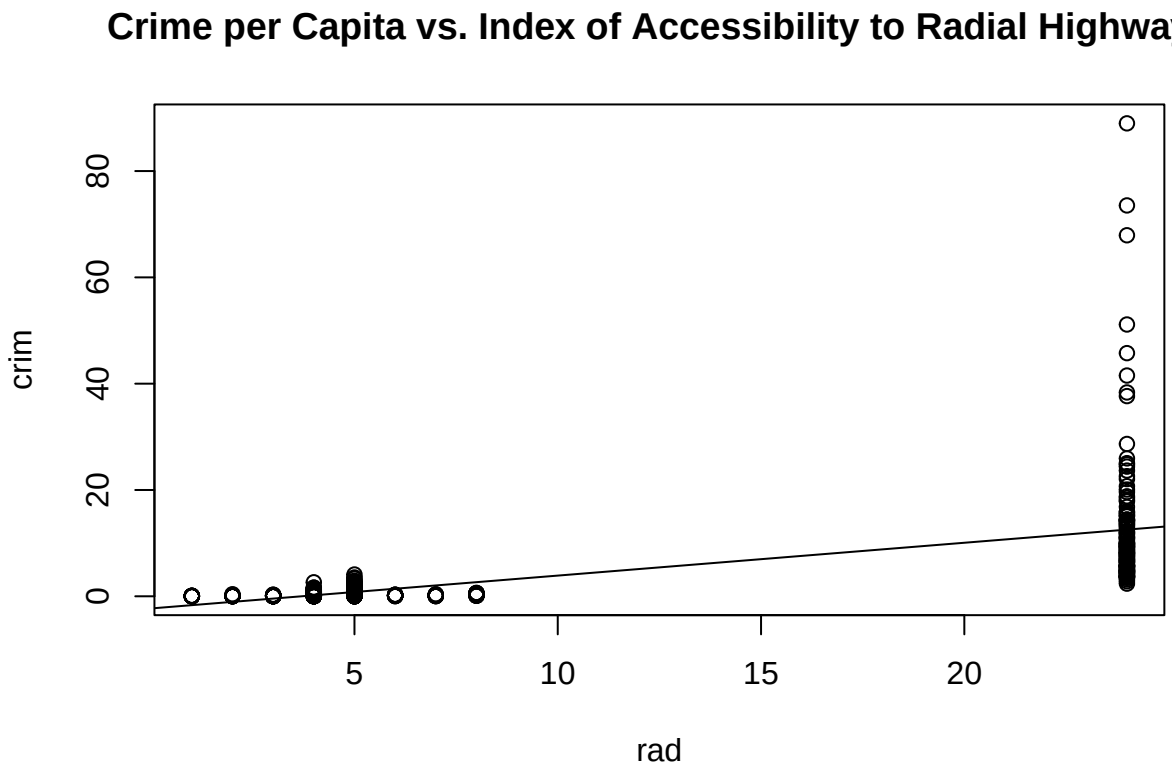
The R^2 for rm (average number of rooms per dwelling) was .048. These R^2 values show that less than half (typically closer to 15%) of the variation could be explained by the predictors.

Based on a combination of p-values and R^2 values, we would expect rad, tax, and lstat to be good predictors. To evaluate this, we can make plots of the data long with residual plots for each predictor

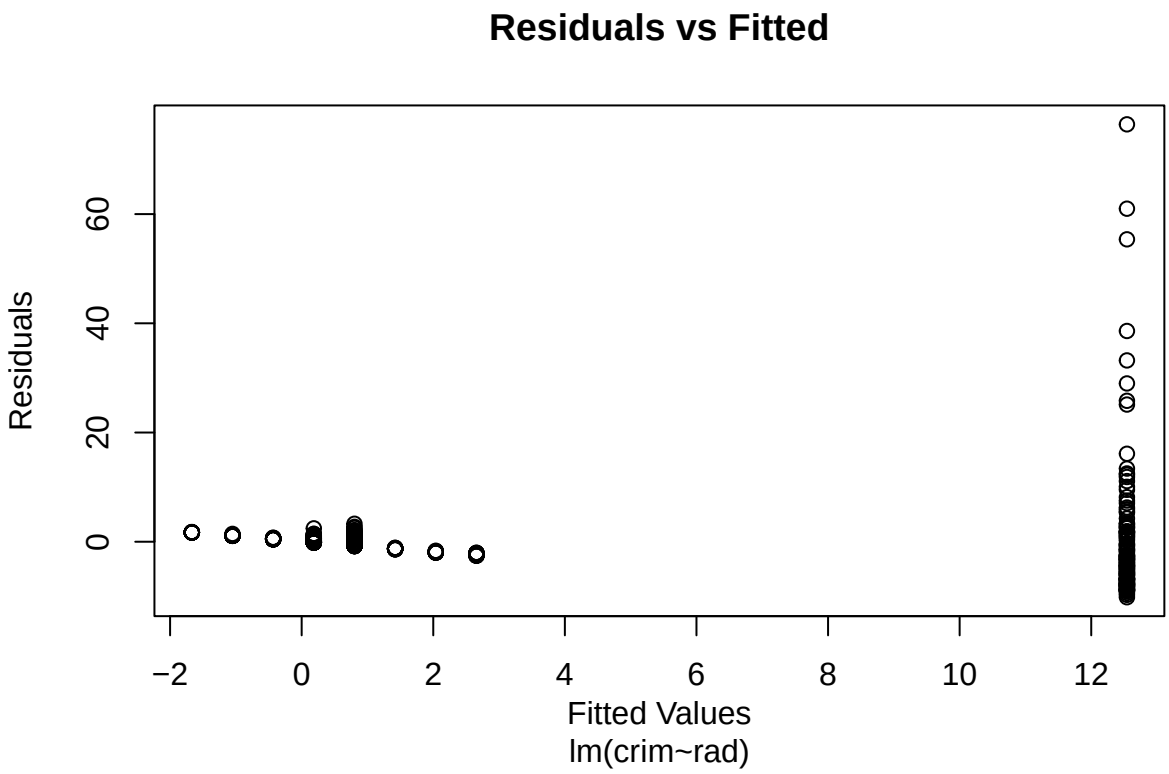
rad

Looking at the plot of crim vs. rad, we see that there is a cluster of values at a rad value of around 25 with much higher crime rates than the the rest of this data-set. This apparent trend that crime is higher in areas that are far from radial highways biases the model upwards. This becomes apparent in the residual plot where we see a downwards linear trend for the residuals outside of the clustered data. This model does not appear suitable for rad to be used as a predictor for crime.

```
plot(rad, crim, main = "Crime per Capita vs. Index of Accessibility to Radial Highways")
abline(rad_lm)
```



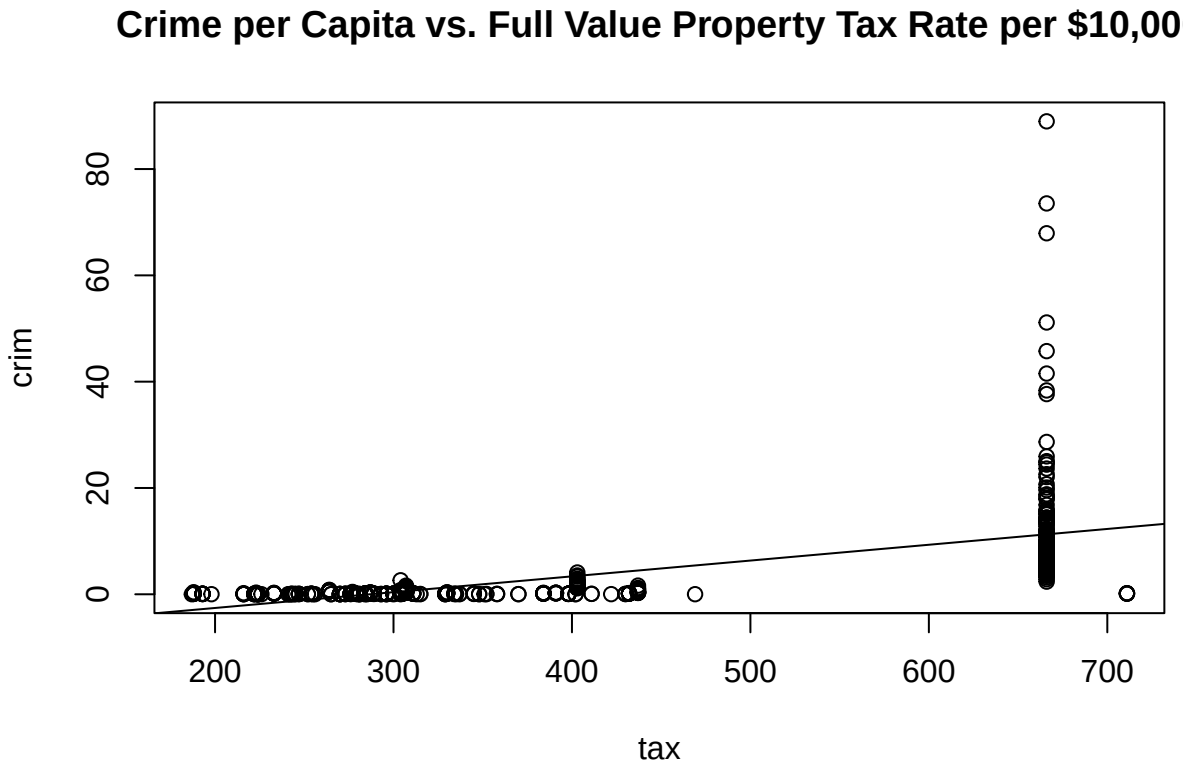

```
plot(rad_lm$fitted.values, rad_lm$residuals,
     xlab = "Fitted Values\nlm(crim~rad)",
     ylab = "Residuals",
     main ="Residuals vs Fitted")
```



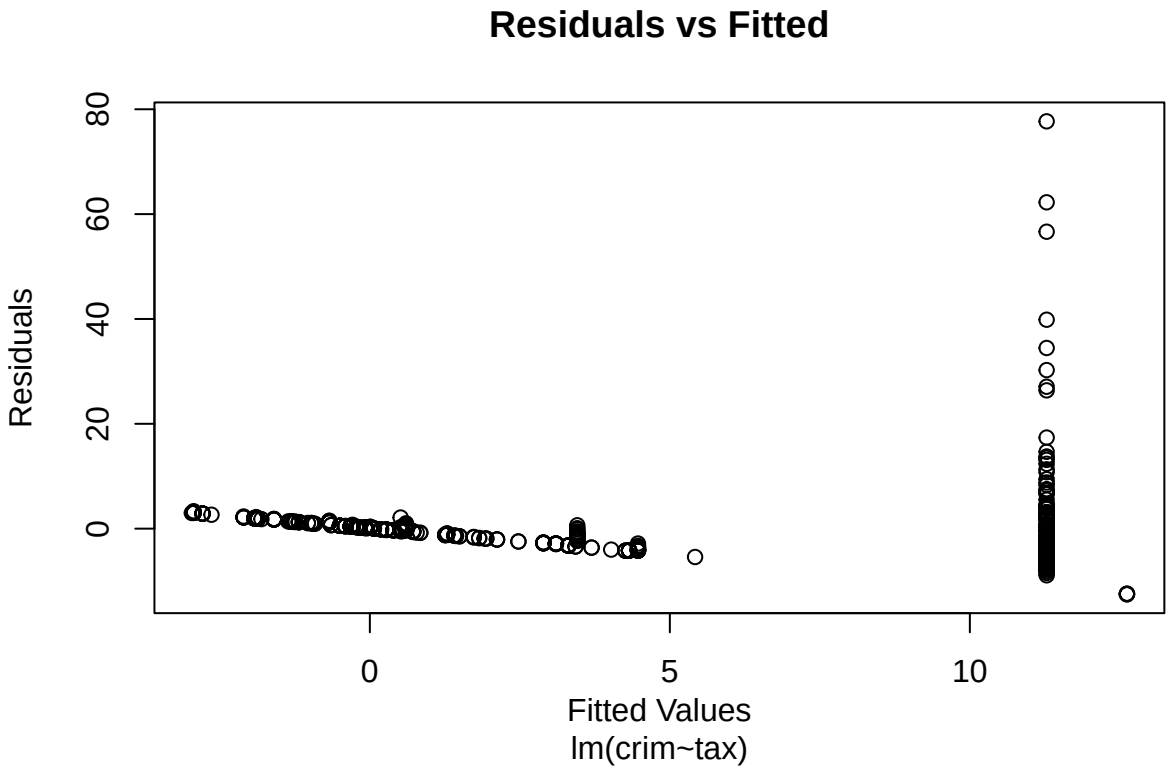
tax

Similarly to rad, when we plot the tax data as a predictor for crime, we see a cluster of data around $670,000 that biases the model. In the residual plot there is a clear linear downward trend in the residuals around the data indicating this bias. This model does not appear suitable for tax to be used as a predictor for crim.

```
plot(tax, crim, main = "Crime per Capita vs. Full Value Property Tax Rate per $10,000" )
abline(tax_lm)
```



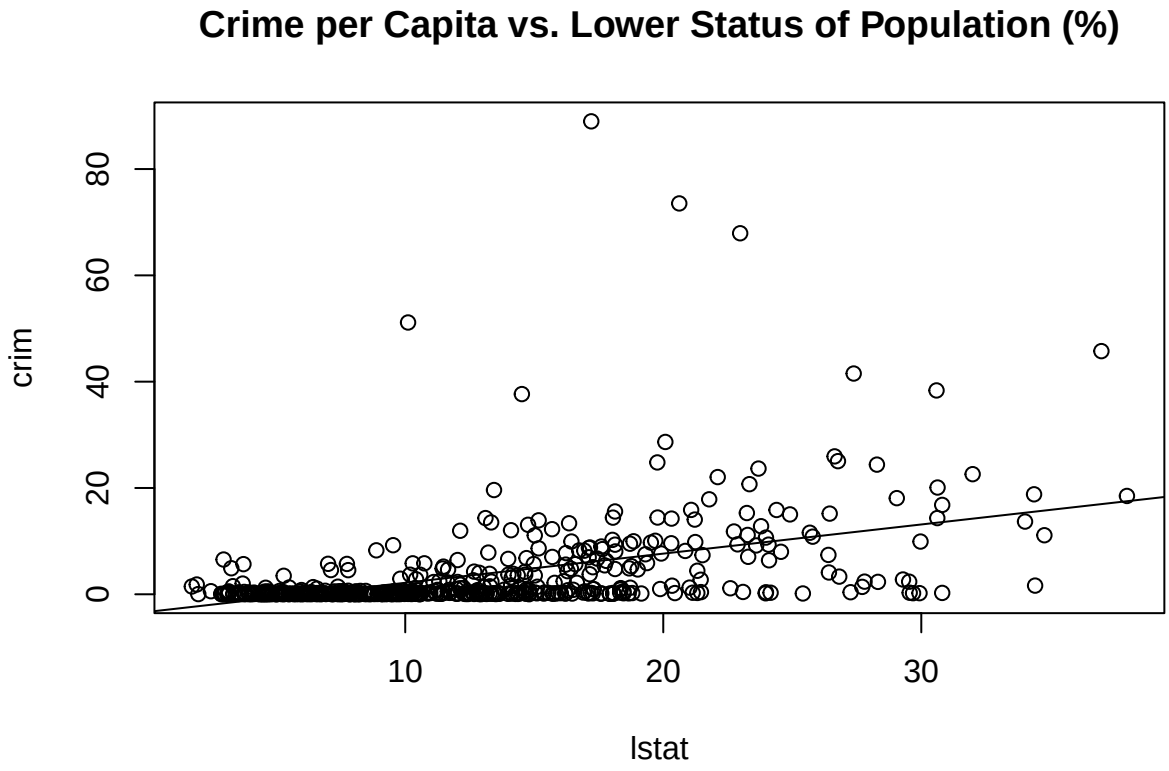

```
plot(tax_lm$fitted.values, tax_lm$residuals,
     xlab = "Fitted Values\nlm(crim~tax)",
     ylab = "Residuals",
     main ="Residuals vs Fitted")
```



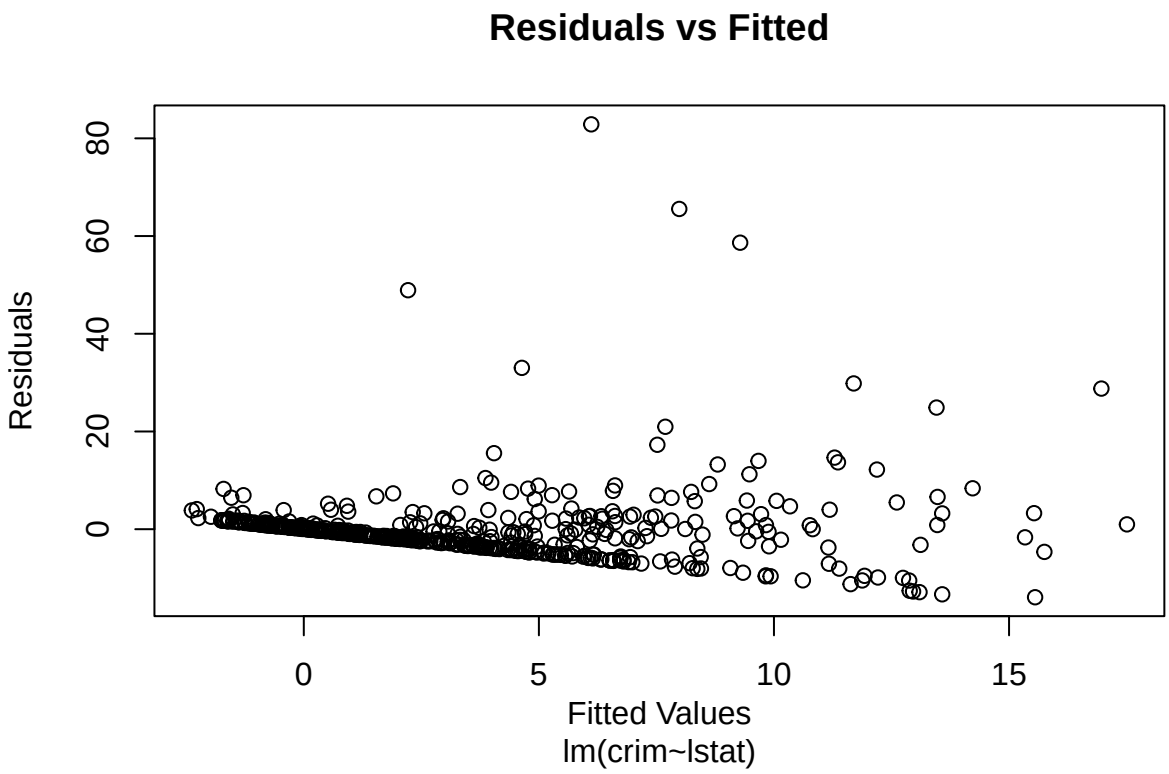
lstat

Looking at the plot for the crim vs. lstat data, we see an apparent trend of crime increasing linearly as the lstat value increases. However, our linear model does not capture the cluster of values around crim = 0 for lower lstat values. There are also a number of outliers in the data. Looking at the residual plot we can see the bias introduced by not capturing the number of data points where crim = 0 as prominent linear downward trend. This model does not appear suitable for lstat to be used as a predictor for crim.

```
plot(lstat, crim, main = "Crime per Capita vs. Lower Status of Population (%)")
abline(lstat_lm)
```




```
plot(lstat_lm$fitted.values, lstat_lm$residuals,
     xlab = "Fitted Values\nlm(crim~lstat)",
     ylab = "Residuals",
     main ="Residuals vs Fitted")
```



3b.

The following creates a multiple linear regression model using all of the predictors to predict crim. Looking at the β values for each predictor, it appears that we can reject the null hypothesis for dis and rad based on the p-values. The medv may also be a statistically significant predictor. These assertions are not based on a p-value of .05, but are based on comparing the p-values among predictors. For instance the p-value fo zn is .017 which may normally be considered significant, but it is several orders of magnitude higher than the dis p-value and many orders of magnitude higher than the rad p-value. Most of the predictors are not statistically significant.

```
multiple_lm = lm(crim ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + black + lstat +
summary(multiple_lm)
```

```
##
## Call:
## lm(formula = crim ~ zn + indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + black + lstat + medv)
##
## 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
```

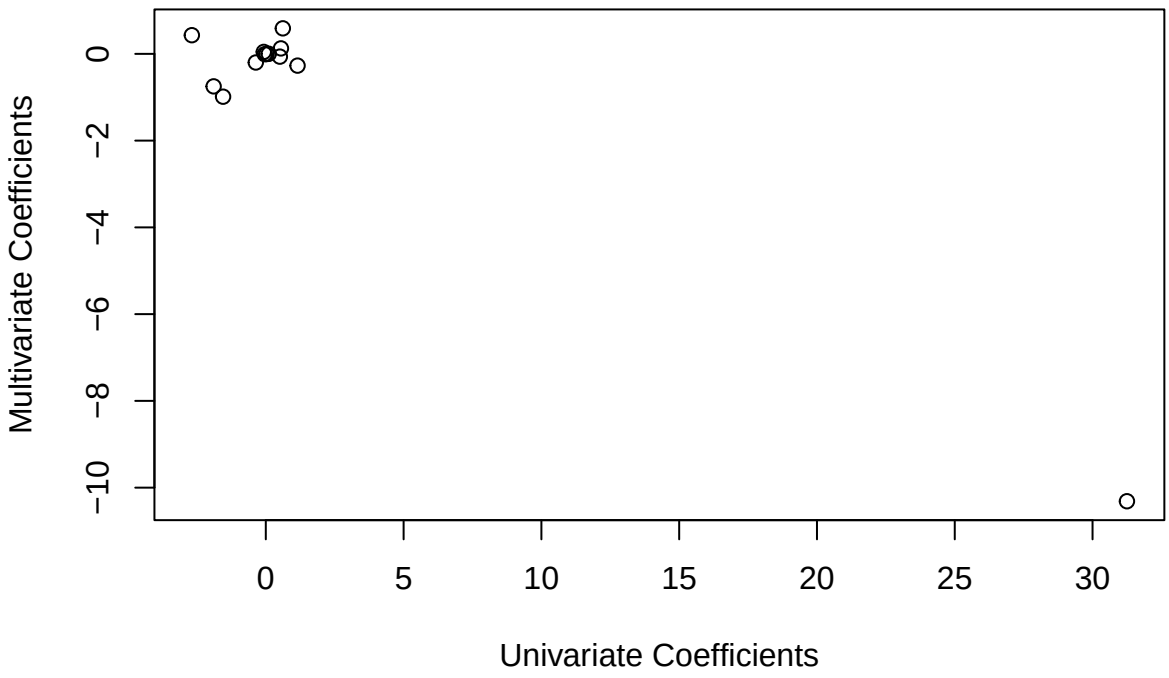
```
## chas         -0.749134   1.180147  -0.635 0.525867
## nox         -10.313535   5.275536  -1.955 0.051152 .
## rm            0.430131   0.612830   0.702 0.483089
## age           0.001452   0.017925   0.081 0.935488
## dis          -0.987176   0.281817  -3.503 0.000502 ***
## rad           0.588209   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
```

3c

When the results of the regression coefficients from the univariate analysis are plotted against the results from the multiple linear regression the most immediate thing to notice is the discrepency in of the values for the nox predictors. At around -10 for the multivariate analysis and over 30 for the univariate analysis it's a clear indicator that the multiple linear regression treats predictors significantly differently. For both analyses the coefficients tend to cluster close to zero, but for the multivariate analysis the values tended to be negative while for the univariate analysis they were more evenly distributed between positive and negative values.

```r
# Create a list of the univariate coefficients
univariate_coef = c(coef(zn_lm)[2], coef(indus_lm)[2], coef(chas_lm)[2], coef(nox_lm)[2],
                    coef(rm_lm)[2], coef(age_lm)[2], coef(dis_lm)[2], coef(rad_lm)[2],
                    coef(tax_lm)[2], coef(ptratio_lm)[2], coef(black_lm)[2],
                    coef(lstat_lm)[2], coef(medv_lm)[2])

# Plot the univariate coefficients against the multivariate coefficients
plot(univariate_coef, coef(multiple_lm)[-1],
     xlab = "Univariate Coefficients",
     ylab = "Multivariate Coefficients")  # The [-1] specifies to ignore the intercept coefficient
```


Multivariate Coefficients
0
−2
−4
−6
−8
−10
0
5
10
15
20
25
30
Univariate Coefficients