Stat 897 Spring 2017 Data Analysis Assignment 3

Penn State

Due September 10, 2017

The goal of this DA assignment will be to familiarize you with linear models and assessing models, as well as beginning to think about model selection.

1. (a) Using the dataset `Auto` from the ISLR package produce simple summaries of the variables in the data. Plot a couple variables against each other where you think one may be a good predictor of the other.

```
set.seed(243)
library(ISLR)
data(Auto)
attach(Auto)
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 
```

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower   
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0  
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0  
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5  
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5  
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0  
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0  
##                                                                 
##      weight      acceleration        year           origin     
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000  
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000  
##  Median :2804   Median :15.50   Median :76.00   Median :1.000  
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577  
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000  
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000  
##                                                                 
##                  name    
##  amc matador       :  5  
##  ford pinto        :  5  
##  toyota corolla    :  5  
##  amc gremlin       :  4  
```

```
##  amc hornet        :  4
##  chevrolet chevette:  4
##  (Other)           :365
```

```r
cor(Auto[1:8])
```

```
##                     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
```

```r
Auto$cylinders <- as.factor(Auto$cylinders)
Auto$year <- as.factor(Auto$year)
Auto$origin <- as.factor(Auto$origin)
```

We see high correlations between

mpg and weight: -0.8322442

mpg and displacement: -0.8051269

mpg and horsepower: -0.7784268

We also see very high correlation between:

displacement and cylinders: 0.9508233

weight and cylinders: 0.8975273

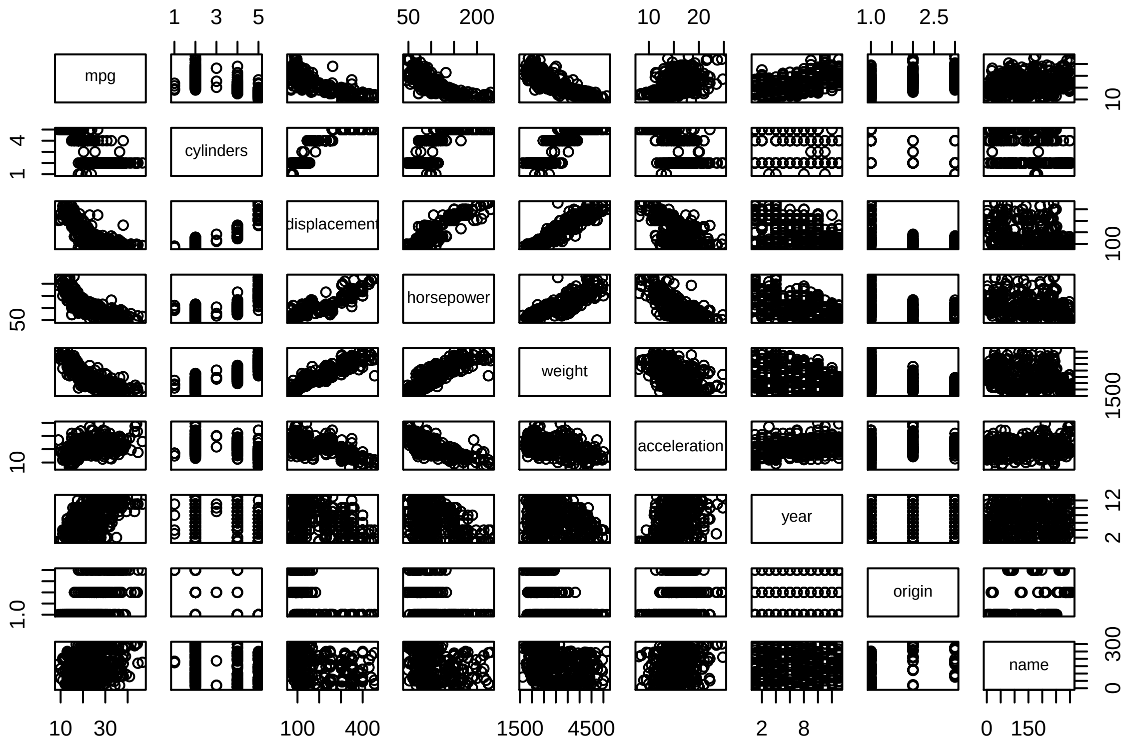
weight and displacement: 0.9329944

weight and horsepower: 0.8645377

displacement and horsepower: 0.8972570

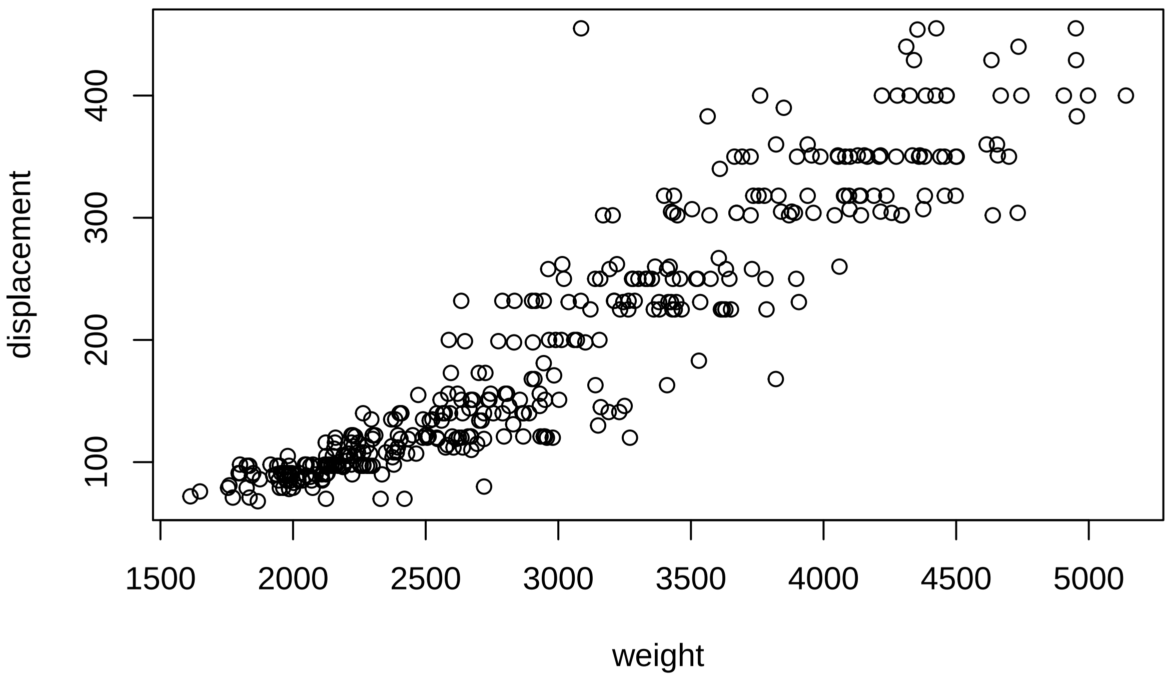
Let’s plot the pairs to validate followed by plotting some of these variables

```r
pairs(Auto)
```

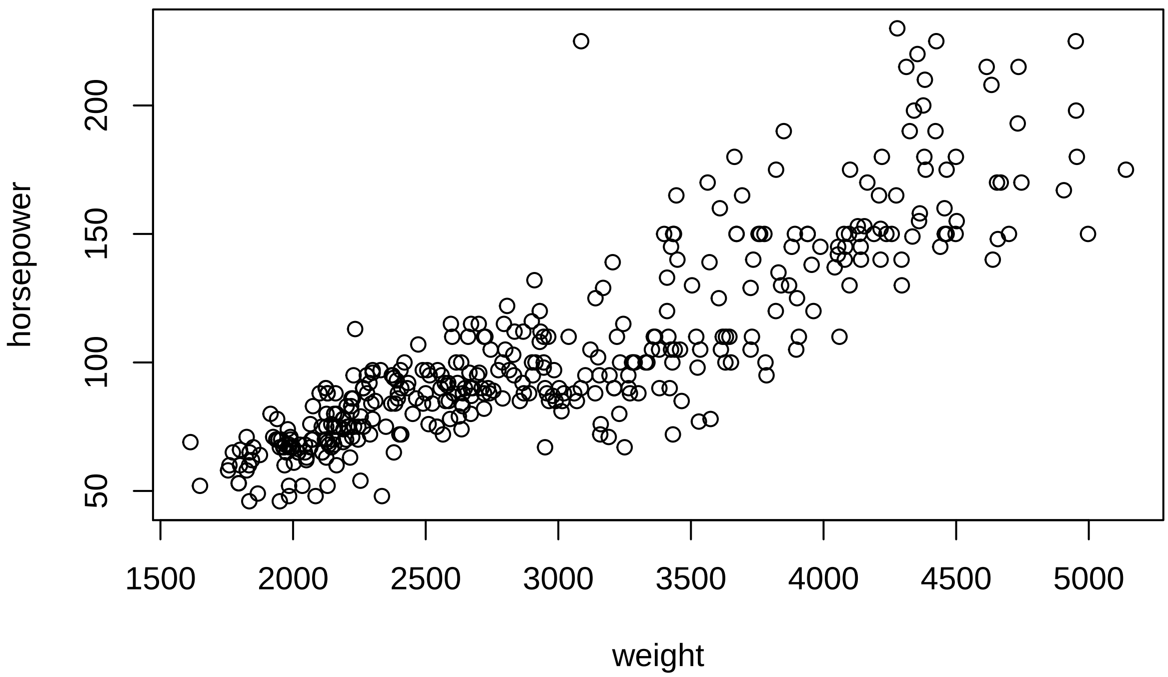


Some of the plots that are of potential interest are:

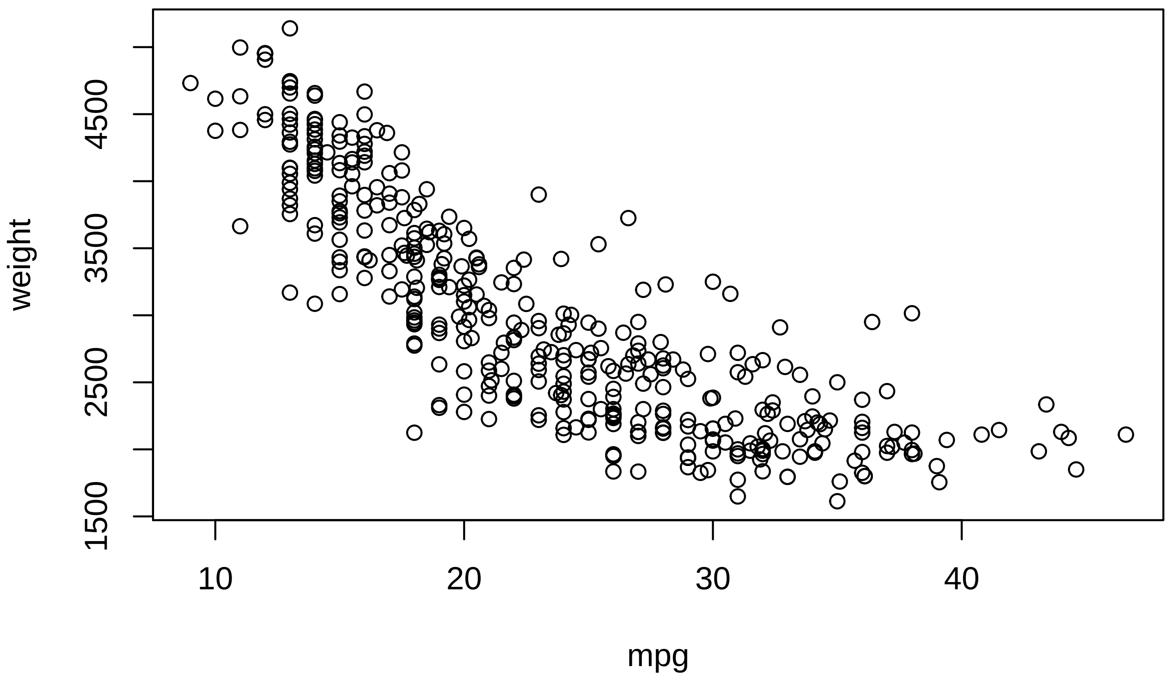
```r
plot(weight, displacement)
```



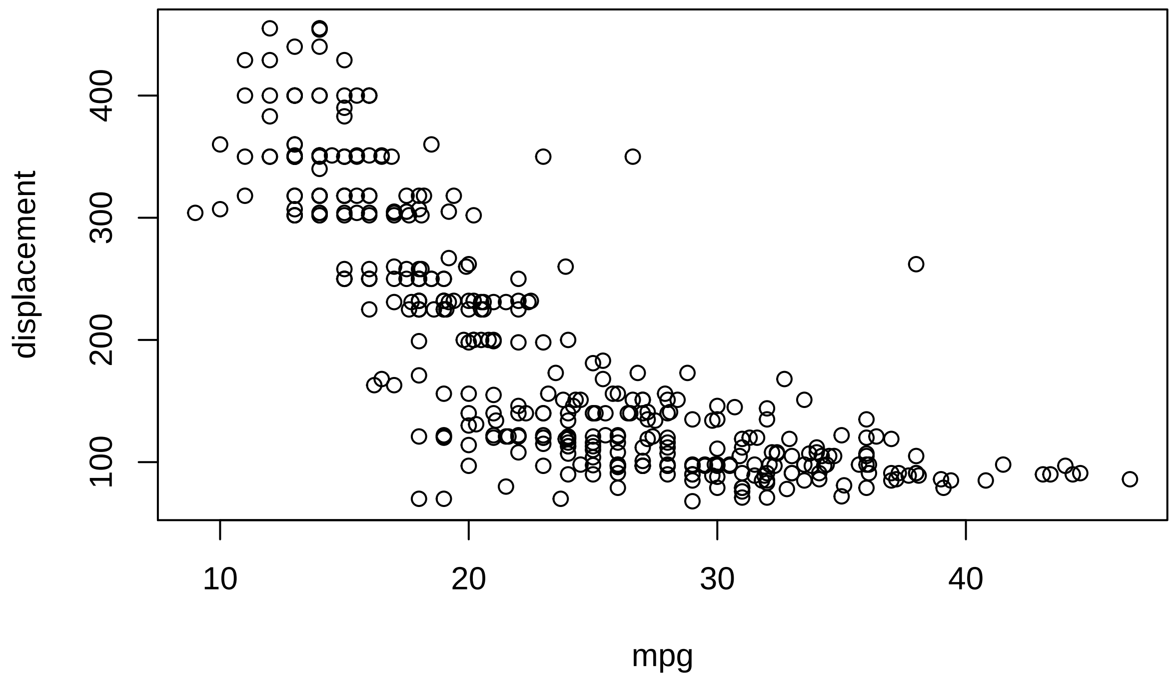

```r
plot(weight, horsepower)
```



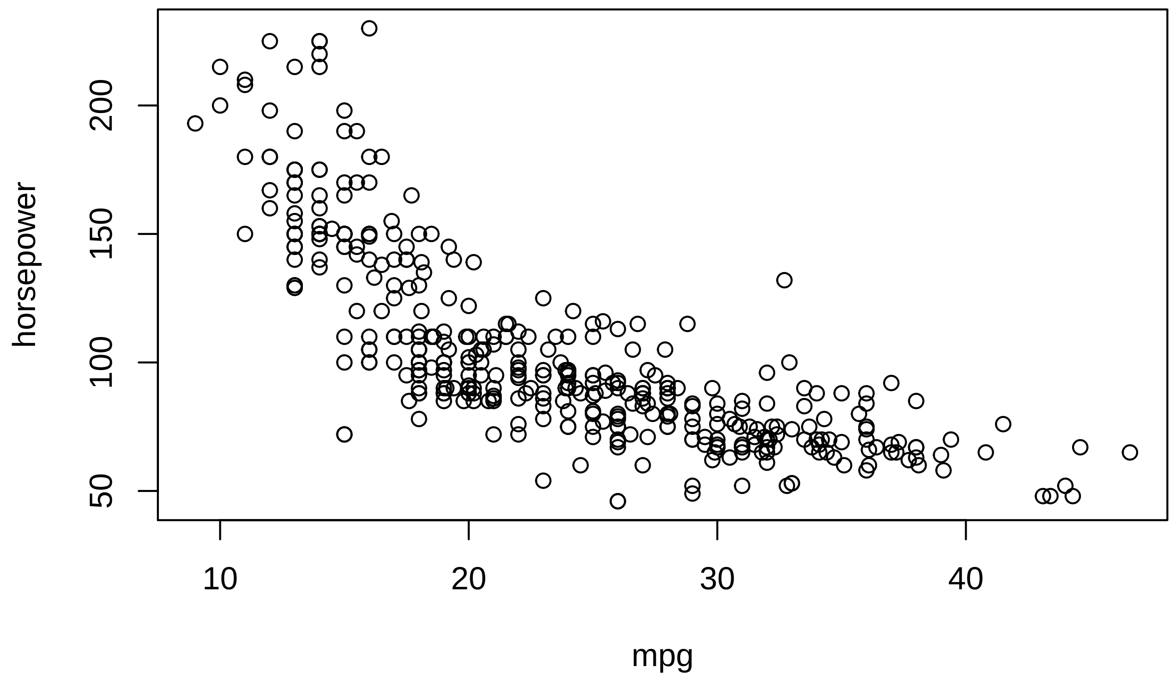

```
plot(mpg, weight)
```




```
plot(mpg, displacement)
```




```
plot(mpg, horsepower)
```



(b) Fit a simple linear model using the two variables you chose and produce a summary of the model.

Let's fit a model between mpg and weight

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

```
##
## Call:
## lm(formula = mpg ~ weight, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9736  -2.7556  -0.3358   2.1379  16.5194
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.216524   0.798673   57.87   <2e-16 ***
## weight      -0.007647   0.000258  -29.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.333 on 390 degrees of freedom
## Multiple R-squared:  0.6926, Adjusted R-squared:  0.6918
## F-statistic: 878.8 on 1 and 390 DF,  p-value: < 2.2e-16
```

(c) Give a 95% confidence interval for the coefficients. Do you think variables are related? Why or why not?

The 95% confidence interval are:

- For the slope it is the estimated coefficient (-0.007647) ± two standard errors (0.000258) = (44.619178, 47.81387)
- For the intercept it is the estimated coefficient (46.216524) ± two standard errors (0.798673) = (-0.008163, -0.007131)

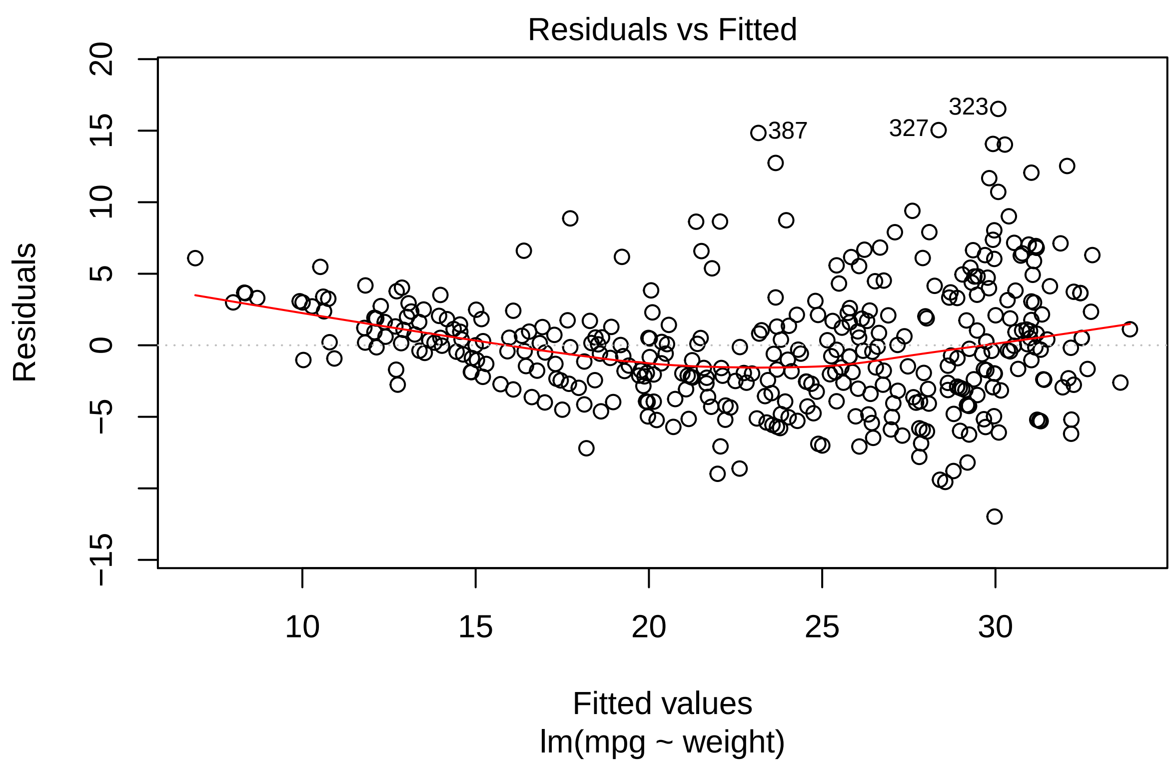
Lets also see how we can generate this using R

```r
confint(lm.fit, level = 0.95)
```

```
##                    2.5 %      97.5 %
## (Intercept) 44.646282308 47.78676679
## weight      -0.008154515 -0.00714017
```

(d) Plot the residuals from your model against the fitted values and comment on anything that looks unusual. (Hint: use the plot.lm function with which = 1.)

```r
plot(lm.fit, which = 1)
```



We can observe the following from the plots:

- Non-constant error variance shows up on a residuals vs. fits - the plot has a "fanning" effect where the residuals are close to 0 for small x values and are more spread out for large x values.

- There is a certain non-linearity in the residuals plot.
- While a few points do look like they are outliers it may be due to the fanning effect of the error variance. Therefore we need further analysis in order to comment about the existence of outliers.

(e) How might you improve your model? (e.g. transformation or addition of a variable).

We can try multiple things as illustrated below: - Transform the predictor (log)

- Add more predictors
- Transform the response variable (log)
- Add more predictors

```
lm.fit1 <- lm(mpg ~ log(weight), data = Auto)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = mpg ~ log(weight), data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.4315  -2.6752  -0.2888   1.9429  16.0136
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 209.9433     6.0002   34.99   <2e-16 ***
## log(weight) -23.4317     0.7534  -31.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.189 on 390 degrees of freedom
## Multiple R-squared:  0.7127, Adjusted R-squared:  0.7119
## F-statistic: 967.3 on 1 and 390 DF,  p-value: < 2.2e-16
```

```
lm.fit2 <- lm(mpg ~ log(weight)+horsepower+displacement, data = Auto)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = mpg ~ log(weight) + horsepower + displacement, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5696  -2.6027  -0.3814   2.1606  15.8216
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  170.341430  14.049315  12.125  < 2e-16 ***
## log(weight)  -17.840770   1.884345  -9.468  < 2e-16 ***
## horsepower    -0.044465   0.012262  -3.626 0.000326 ***
## displacement  -0.001298   0.006122  -0.212 0.832192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4.091 on 388 degrees of freedom
## Multiple R-squared:  0.7273, Adjusted R-squared:  0.7252
## F-statistic:   345 on 3 and 388 DF,  p-value: < 2.2e-16
```

```r
lm.fit3 <- lm(log(mpg) ~ log(weight), data = Auto)
summary(lm.fit3)
```

```
##
## Call:
## lm(formula = log(mpg) ~ log(weight), data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52321 -0.10446 -0.00772  0.10124  0.59445
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.5152     0.2365   48.69   <2e-16 ***
## log(weight)  -1.0575     0.0297  -35.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1651 on 390 degrees of freedom
## Multiple R-squared:  0.7648, Adjusted R-squared:  0.7642
## F-statistic:  1268 on 1 and 390 DF,  p-value: < 2.2e-16
```

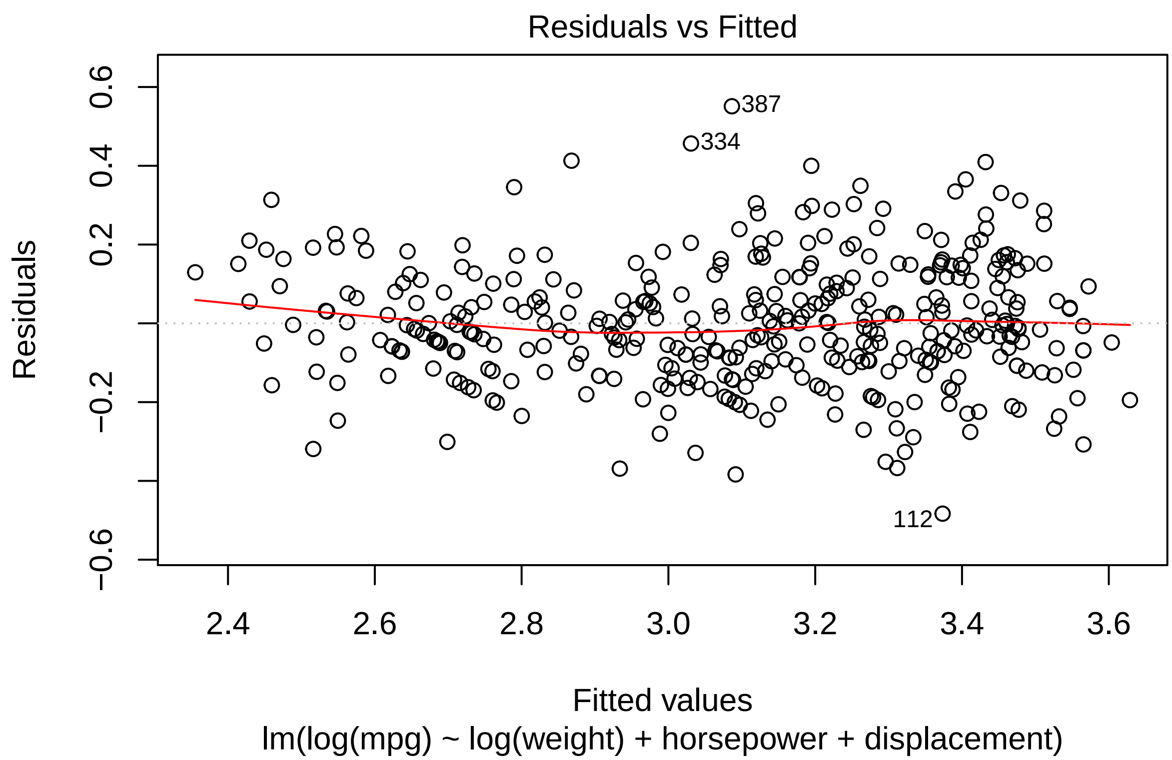
```r
lm.fit4 <- lm(log(mpg) ~ log(weight)+horsepower+displacement, data = Auto)
summary(lm.fit4)
```

```
##
## Call:
## lm(formula = log(mpg) ~ log(weight) + horsepower + displacement,
##     data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48317 -0.09662 -0.00817  0.10240  0.55110
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.6460294  0.5308616  16.287  < 2e-16 ***
## log(weight)  -0.6567408  0.0712011  -9.224  < 2e-16 ***
## horsepower   -0.0024000  0.0004633  -5.180 3.58e-07 ***
## displacement -0.0003594  0.0002313  -1.554    0.121
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1546 on 388 degrees of freedom
## Multiple R-squared:  0.7949, Adjusted R-squared:  0.7933
## F-statistic: 501.2 on 3 and 388 DF,  p-value: < 2.2e-16
```

We see from the above that in all the steps the model fitted better to the training data. We saw the RSE go down and R2 (also adjusted R2) climb up.

Lets check the residuals for the new model.

```r
plot(lm.fit4, which = 1)
```



We see from the above that the model properties have improved (still room for improvement) in terms addressing the non-linearity and non constant variance seen before.

2. (a) Load the `mlbench` library and upload the `BostonHousing` data. Produce a summary of the variable `medv`.

```r
#install.packages('mlbench')
library(mlbench)
data("BostonHousing")
attach(BostonHousing)
```

(b) Using `medv` as your response variable fit a linear regression model with all othe variables as predictors. Compute the training MSE.

```r
lm.fit = lm(medv~., data=BostonHousing)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ ., data = BostonHousing)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas1        2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## b            9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```r
lm.predict <- predict(lm.fit)

# 2 diff ways to get to the MSE
mean((BostonHousing$medv - lm.predict)^2)
```

```
## [1] 21.89483
```

```r
mse <- function(model)
    mean(model$residuals^2)

mse(lm.fit)
```

```
## [1] 21.89483
```

Linear regression MSE = 21.89483

(c) Now find a good model for predicting `medv`. Explain your process in choosing the model and why it is a good prediction model. Feel free to use any number of the other variables in the data as predictors.

Let's start from where we left in part b of using all the other variables. We will use individual names instead of . notation

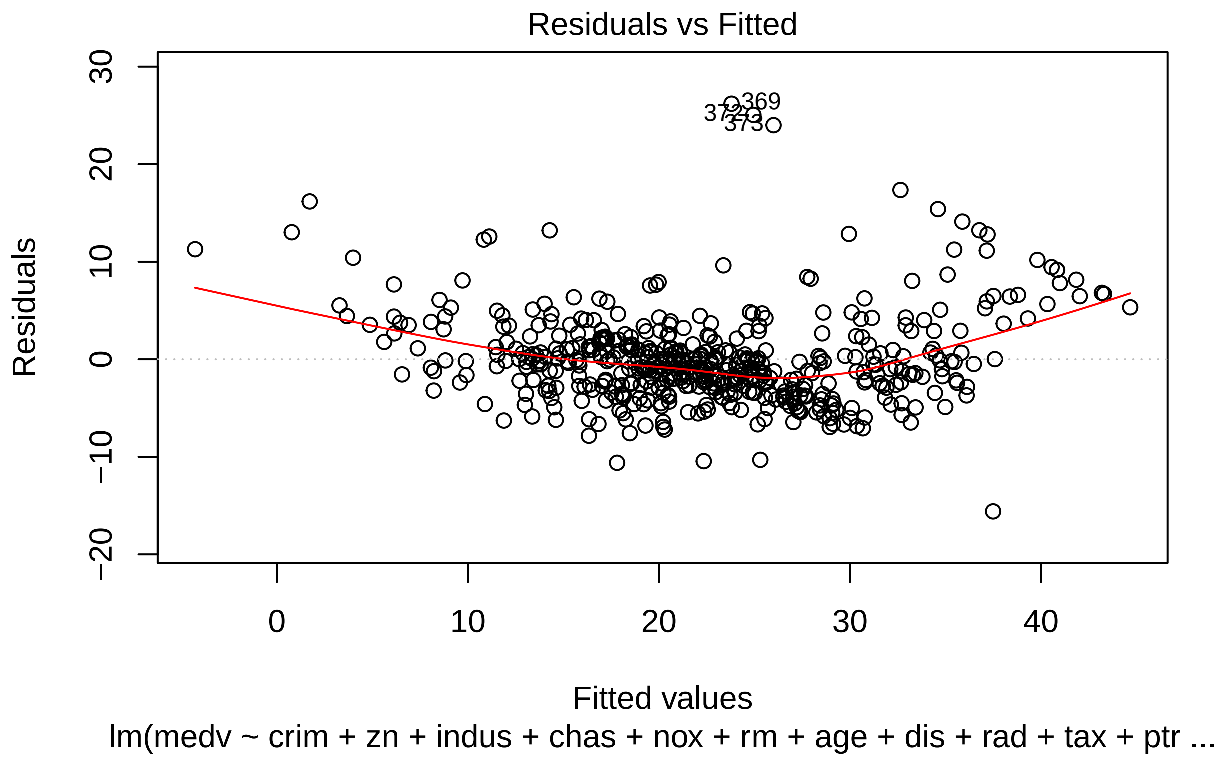
The first step is the same as part b. We plot the residuals.

```r
lm.fit1 = lm(medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+b+lstat, data=BostonHousing)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + indus + chas + nox + rm + age +
```

```
##     dis + rad + tax + ptratio + b + lstat, data = BostonHousing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas1        2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## b            9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```r
plot(lm.fit1, which = 1)
```



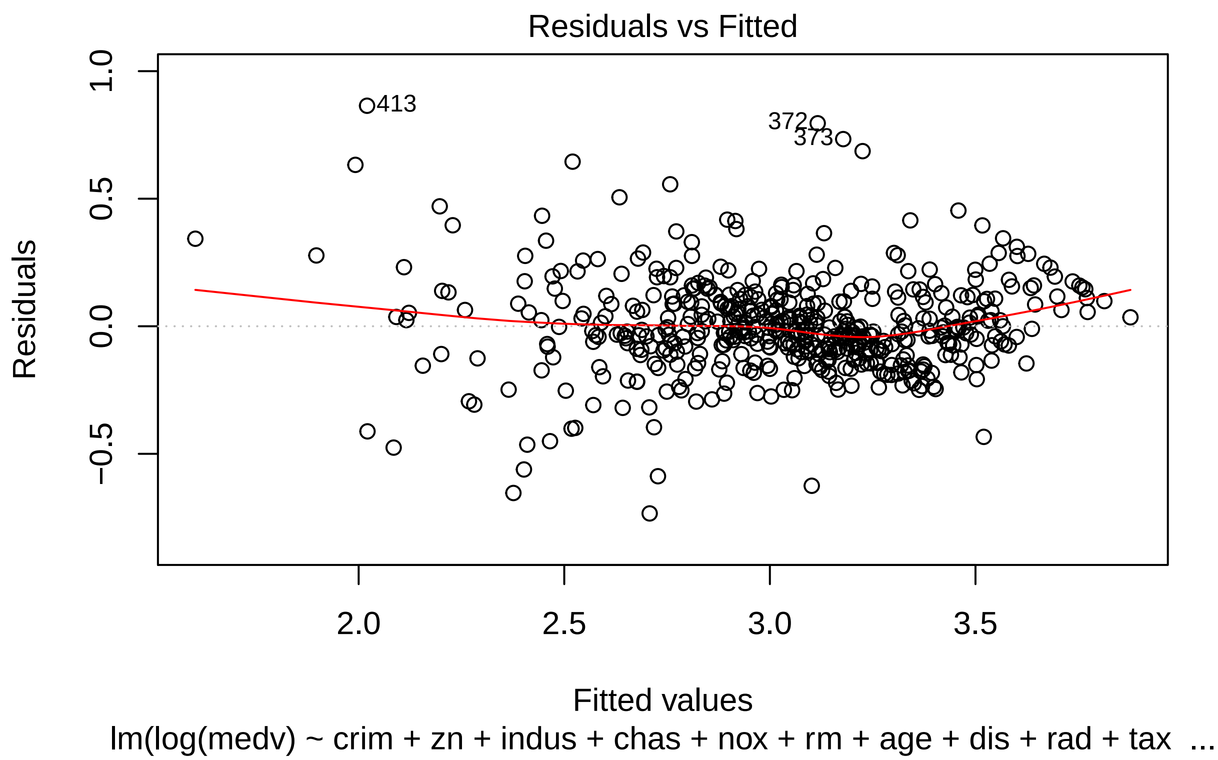
We observe some non-linearity and so use log transformation of the response variable

```
lm.fit1 = lm(log(medv)~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+b+lstat, data=BostonHousing)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = log(medv) ~ crim + zn + indus + chas + nox + rm +
##     age + dis + rad + tax + ptratio + b + lstat, data = BostonHousing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73361 -0.09747 -0.01657  0.09629  0.86435
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.1020423  0.2042726  20.081  < 2e-16 ***
## crim        -0.0102715  0.0013155  -7.808 3.52e-14 ***
## zn           0.0011725  0.0005495   2.134 0.033349 *
## indus        0.0024668  0.0024614   1.002 0.316755
## chas1        0.1008876  0.0344859   2.925 0.003598 **
## nox         -0.7783993  0.1528902  -5.091 5.07e-07 ***
## rm           0.0908331  0.0167280   5.430 8.87e-08 ***
## age          0.0002106  0.0005287   0.398 0.690567
## dis         -0.0490873  0.0079834  -6.149 1.62e-09 ***
## rad          0.0142673  0.0026556   5.373 1.20e-07 ***
## tax         -0.0006258  0.0001505  -4.157 3.80e-05 ***
```

```
## ptratio     -0.0382715  0.0052365  -7.309 1.10e-12 ***
## b            0.0004136  0.0001075   3.847 0.000135 ***
## lstat       -0.0290355  0.0020299 -14.304  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1899 on 492 degrees of freedom
## Multiple R-squared:  0.7896, Adjusted R-squared:  0.7841
## F-statistic: 142.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
plot(lm.fit1, which = 1)
```



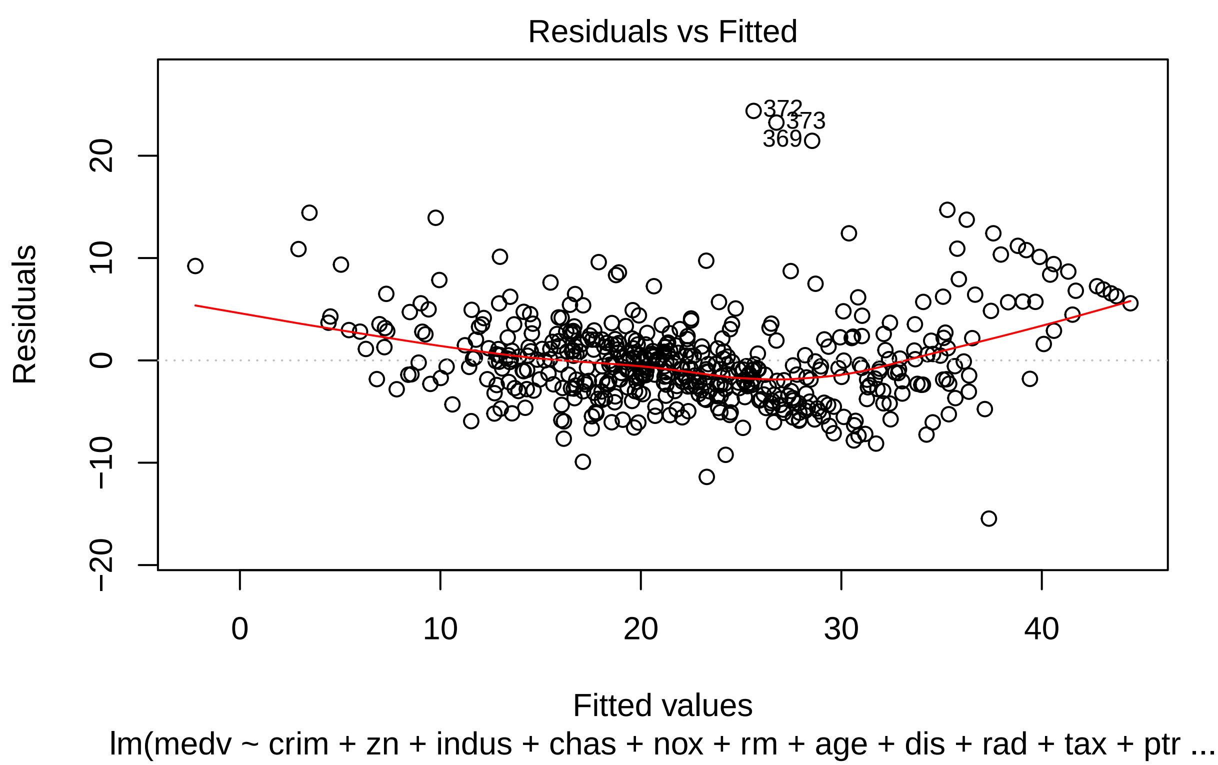
We still see non-linearity. Lets further add sqrt transformation of lstat

```
lm.fit1 = lm(medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+b+sqrt(lstat), data=BostonHousing)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + indus + chas + nox + rm + age +
##     dis + rad + tax + ptratio + b + sqrt(lstat), data = BostonHousing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.4606  -2.5433  -0.5706   1.9341  24.3759
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  49.059990   5.059604   9.696  < 2e-16 ***
## crim         -0.112055   0.030864  -3.631 0.000312 ***
## zn            0.038129   0.012939   2.947 0.003364 **
## indus         0.028333   0.058029   0.488 0.625586
## chas1         2.423383   0.813956   2.977 0.003051 **
## nox         -16.815531   3.607294  -4.662 4.05e-06 ***
## rm            2.987892   0.406912   7.343 8.73e-13 ***
## age           0.017877   0.012647   1.414 0.158128
## dis          -1.361803   0.188676  -7.218 2.02e-12 ***
## rad           0.306544   0.062612   4.896 1.33e-06 ***
## tax          -0.011956   0.003549  -3.369 0.000814 ***
## ptratio      -0.903300   0.123637  -7.306 1.12e-12 ***
## b             0.008097   0.002538   3.190 0.001513 **
## sqrt(lstat)  -4.918619   0.366222 -13.431  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.479 on 492 degrees of freedom
## Multiple R-squared:  0.7689, Adjusted R-squared:  0.7628
## F-statistic: 125.9 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
plot(lm.fit1, which = 1)
```

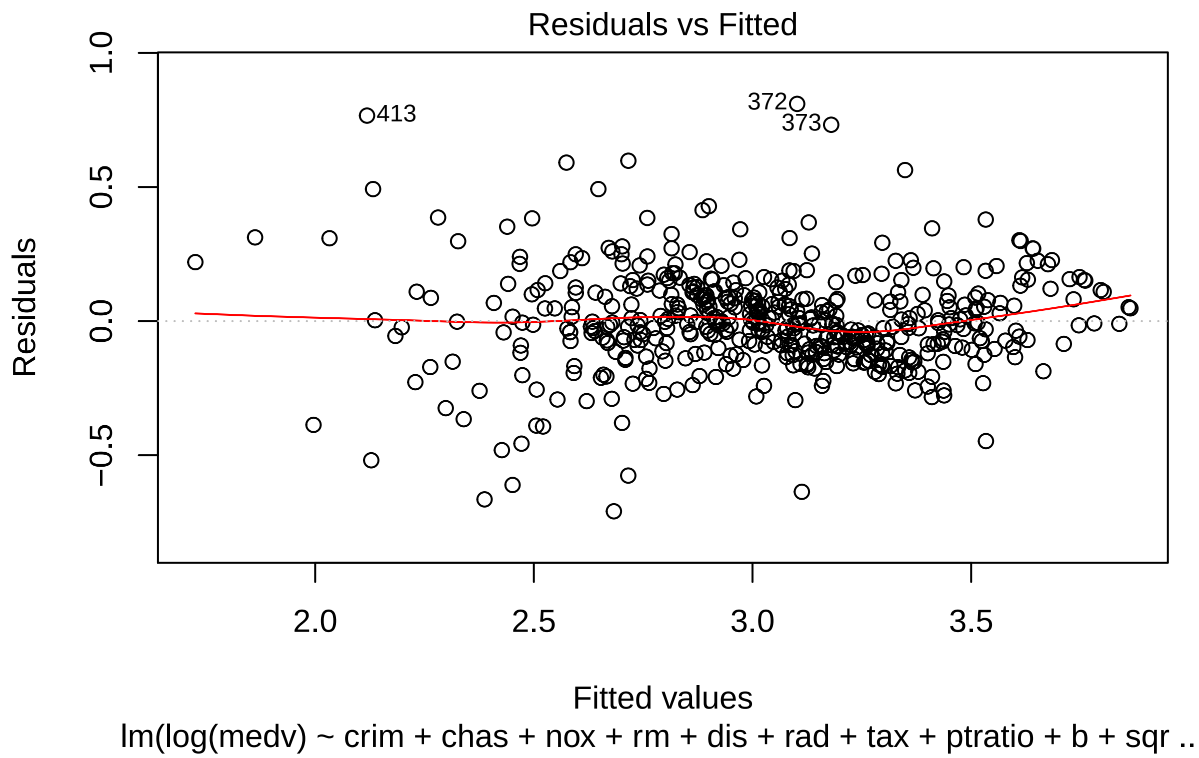


Remove the predictors that are not significant in the result.

```
lm.fit1 = lm(log(medv)~crim+chas+nox+rm+dis+rad+tax+ptratio+b+sqrt(lstat), data=BostonHousing)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = log(medv) ~ crim + chas + nox + rm + dis + rad +
##     tax + ptratio + b + sqrt(lstat), data = BostonHousing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70892 -0.09659 -0.01160  0.09713  0.80989
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.5635403  0.2051498  22.245  < 2e-16 ***
## crim        -0.0107087  0.0012616  -8.488 2.46e-16 ***
## chas1        0.0978987  0.0332245   2.947 0.003365 **
## nox         -0.6713329  0.1373404  -4.888 1.38e-06 ***
## rm           0.0729594  0.0160778   4.538 7.14e-06 ***
## dis         -0.0451202  0.0062549  -7.214 2.06e-12 ***
## rad          0.0129082  0.0024588   5.250 2.26e-07 ***
## tax         -0.0005021  0.0001286  -3.905 0.000107 ***
## ptratio     -0.0370226  0.0047378  -7.814 3.34e-14 ***
## b            0.0003910  0.0001040   3.759 0.000191 ***
## sqrt(lstat) -0.2301208  0.0138899 -16.567  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1841 on 495 degrees of freedom
## Multiple R-squared:  0.8011, Adjusted R-squared:  0.7971
## F-statistic: 199.4 on 10 and 495 DF,  p-value: < 2.2e-16
```

```r
plot(lm.fit1, which = 1)
```



Let's calculate the MSE of this model:

```
lm.predict1 = exp(predict(lm.fit1))
mean((BostonHousing$medv - lm.predict1)^2)
```

```
## [1] 17.10186
```

MSE is 17.10186

Another interesting tweak to the model that is not linear progression from the above improvements is to include all the predictors and their pairwise interaction terms:

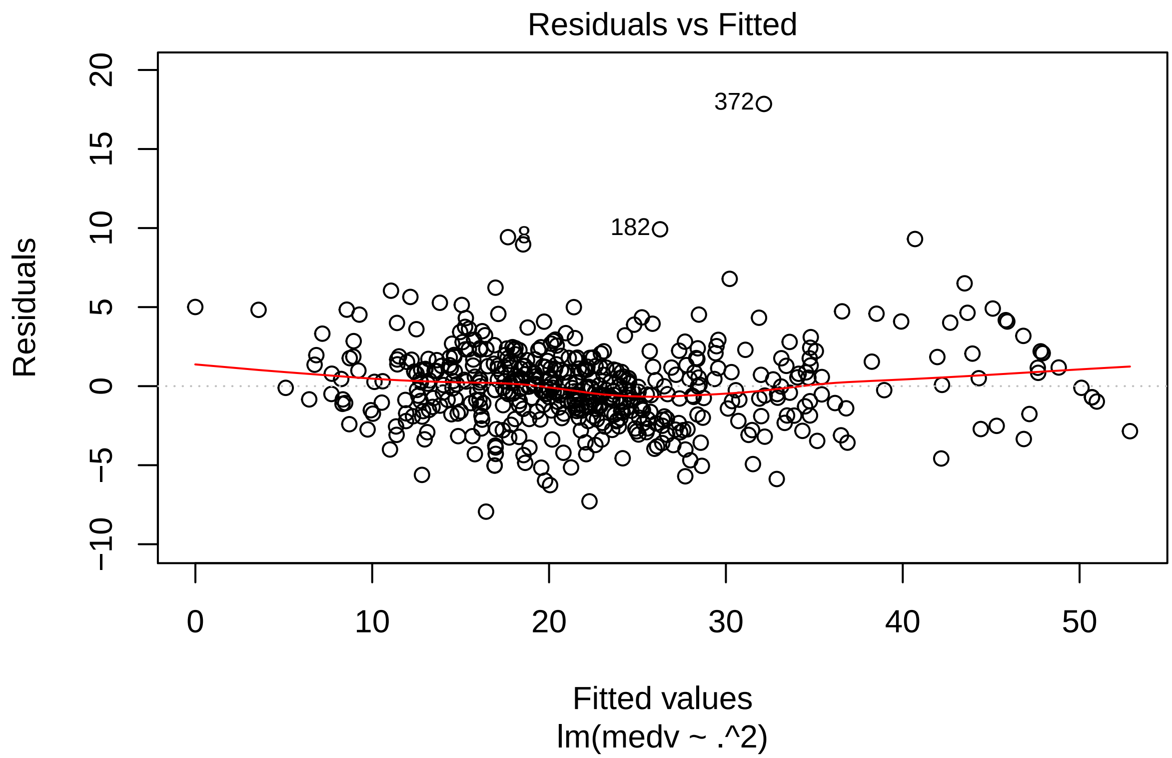
```
lm.fit2 = lm(medv~.^2, data=BostonHousing)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ .^2, data = BostonHousing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9374 -1.5344 -0.1068  1.2973 17.8500
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.579e+02  6.800e+01  -2.323 0.020683 *
## crim          -1.707e+01  6.554e+00  -2.605 0.009526 **
## zn            -7.529e-02  4.580e-01  -0.164 0.869508
## indus         -2.819e+00  1.696e+00  -1.663 0.097111 .
```

```
## chas1           4.451e+01  1.952e+01   2.280 0.023123 *
## nox             2.006e+01  7.516e+01   0.267 0.789717
## rm              2.527e+01  5.699e+00   4.435 1.18e-05 ***
## age             1.263e+00  2.728e-01   4.630 4.90e-06 ***
## dis            -1.698e+00  4.604e+00  -0.369 0.712395
## rad             1.861e+00  2.464e+00   0.755 0.450532
## tax             3.670e-02  1.440e-01   0.255 0.798978
## ptratio         2.725e+00  2.850e+00   0.956 0.339567
## b               9.942e-02  7.468e-02   1.331 0.183833
## lstat           1.656e+00  8.533e-01   1.940 0.053032 .
## crim:zn         4.144e-01  1.804e-01   2.297 0.022128 *
## crim:indus     -4.693e-02  4.480e-01  -0.105 0.916621
## crim:chas1      2.428e+00  5.710e-01   4.251 2.63e-05 ***
## crim:nox       -1.108e+00  9.285e-01  -1.193 0.233425
## crim:rm         2.163e-01  4.907e-02   4.409 1.33e-05 ***
## crim:age       -3.083e-03  3.781e-03  -0.815 0.415315
## crim:dis       -1.903e-01  1.060e-01  -1.795 0.073307 .
## crim:rad       -6.584e-01  5.815e-01  -1.132 0.258198
## crim:tax        3.479e-02  4.287e-02   0.812 0.417453
## crim:ptratio    4.915e-01  3.328e-01   1.477 0.140476
## crim:b         -4.612e-04  1.793e-04  -2.572 0.010451 *
## crim:lstat      2.964e-02  6.544e-03   4.530 7.72e-06 ***
## zn:indus       -6.731e-04  4.651e-03  -0.145 0.885000
## zn:chas1       -5.230e-02  6.450e-02  -0.811 0.417900
## zn:nox          1.998e-03  4.721e-01   0.004 0.996625
## zn:rm          -7.286e-04  2.602e-02  -0.028 0.977672
## zn:age         -1.249e-06  8.514e-04  -0.001 0.998830
## zn:dis          1.097e-02  7.550e-03   1.452 0.147121
## zn:rad         -3.200e-03  6.975e-03  -0.459 0.646591
## zn:tax          3.937e-04  1.783e-04   2.209 0.027744 *
## zn:ptratio     -4.578e-03  7.015e-03  -0.653 0.514325
## zn:b            1.159e-04  7.599e-04   0.153 0.878841
## zn:lstat       -1.064e-02  4.662e-03  -2.281 0.023040 *
## indus:chas1    -3.672e-01  3.780e-01  -0.971 0.331881
## indus:nox       3.138e+00  1.449e+00   2.166 0.030855 *
## indus:rm        3.301e-01  1.327e-01   2.488 0.013257 *
## indus:age      -4.865e-04  3.659e-03  -0.133 0.894284
## indus:dis      -4.486e-02  6.312e-02  -0.711 0.477645
## indus:rad      -2.089e-02  5.020e-02  -0.416 0.677560
## indus:tax       3.129e-04  6.034e-04   0.519 0.604322
## indus:ptratio -6.011e-02  3.783e-02  -1.589 0.112820
## indus:b         1.122e-03  2.034e-03   0.552 0.581464
## indus:lstat     5.063e-03  1.523e-02   0.332 0.739789
## chas1:nox      -3.272e+01  1.243e+01  -2.631 0.008820 **
## chas1:rm       -5.384e+00  1.150e+00  -4.681 3.87e-06 ***
## chas1:age       3.040e-02  5.840e-02   0.521 0.602982
## chas1:dis       9.022e-01  1.334e+00   0.676 0.499143
## chas1:rad      -7.773e-01  5.707e-01  -1.362 0.173907
## chas1:tax       4.627e-02  3.645e-02   1.270 0.204930
## chas1:ptratio -6.145e-01  6.914e-01  -0.889 0.374604
## chas1:b         2.500e-02  1.567e-02   1.595 0.111423
## chas1:lstat    -2.980e-01  1.845e-01  -1.615 0.107008
## nox:rm          5.990e+00  5.468e+00   1.095 0.273952
## nox:age        -7.273e-01  2.340e-01  -3.108 0.002012 **
```

```
## nox:dis        5.694e+00  3.723e+00   1.529 0.126969
## nox:rad       -1.994e-01  1.897e+00  -0.105 0.916360
## nox:tax       -2.793e-02  1.312e-01  -0.213 0.831559
## nox:ptratio   -3.669e+00  3.096e+00  -1.185 0.236648
## nox:b         -1.854e-02  3.615e-02  -0.513 0.608298
## nox:lstat      1.119e+00  6.511e-01   1.719 0.086304 .
## rm:age        -6.277e-02  2.203e-02  -2.849 0.004606 **
## rm:dis         3.190e-01  3.295e-01   0.968 0.333516
## rm:rad        -8.422e-02  1.527e-01  -0.552 0.581565
## rm:tax        -2.242e-02  9.910e-03  -2.262 0.024216 *
## rm:ptratio    -4.880e-01  2.172e-01  -2.247 0.025189 *
## rm:b          -4.528e-03  3.351e-03  -1.351 0.177386
## rm:lstat      -2.968e-01  4.316e-02  -6.878 2.24e-11 ***
## age:dis       -1.678e-02  8.882e-03  -1.889 0.059589 .
## age:rad        1.442e-02  4.212e-03   3.423 0.000682 ***
## age:tax       -3.403e-04  2.187e-04  -1.556 0.120437
## age:ptratio   -7.520e-03  6.793e-03  -1.107 0.268946
## age:b         -7.029e-04  2.136e-04  -3.291 0.001083 **
## age:lstat     -6.023e-03  1.936e-03  -3.111 0.001991 **
## dis:rad       -5.580e-02  7.075e-02  -0.789 0.430678
## dis:tax       -3.882e-03  2.496e-03  -1.555 0.120623
## dis:ptratio   -4.786e-02  9.983e-02  -0.479 0.631920
## dis:b         -5.194e-03  5.541e-03  -0.937 0.349116
## dis:lstat      1.350e-01  4.866e-02   2.775 0.005774 **
## rad:tax        3.131e-05  1.446e-03   0.022 0.982729
## rad:ptratio   -4.379e-02  8.392e-02  -0.522 0.602121
## rad:b         -4.362e-04  2.518e-03  -0.173 0.862561
## rad:lstat     -2.529e-02  1.816e-02  -1.392 0.164530
## tax:ptratio    7.854e-03  2.504e-03   3.137 0.001830 **
## tax:b         -4.785e-07  1.999e-04  -0.002 0.998091
## tax:lstat     -1.403e-03  1.208e-03  -1.162 0.245940
## ptratio:b      1.203e-03  3.361e-03   0.358 0.720508
## ptratio:lstat  3.901e-03  2.985e-02   0.131 0.896068
## b:lstat       -6.118e-04  4.157e-04  -1.472 0.141837
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.852 on 414 degrees of freedom
## Multiple R-squared:  0.9212, Adjusted R-squared:  0.9039
## F-statistic: 53.18 on 91 and 414 DF,  p-value: < 2.2e-16
```

```r
plot(lm.fit2, which = 1)
```



The issue here is that while the R2 has improved there are a lot of parameters that are not significant and can be dropped from the model. For our discussion here the model previous to the last will be used for following exercises i.e.

lm.fit1 = lm(log(medv)~crim+chas+nox+rm+dis+rad+tax+ptratio+b+sqrt(lstat), data=BostonHousing)

(d) Now let's briefly consider a comparison of neural networks with linear regression. We will use the `nnet` package and the function `nnet`. Using the same data fit a neural network with `medv` as response and all other variables as predictors.

```r
library(nnet)
```

(Note: for the neural network you need to scale the response variable so that all values are between 0 and 1. An easy way to do this is by dividing by the maximum value. When you predict values remember to restore the original scale.)

```r
medv_max = max(medv)
medv_max
```

```
## [1] 50
```

Since the max for response variable medv is 50, we will divide by this number to scale the variable between 0 and 1

The model is:

```r
# Since we are using nnet to perform regression (rather than a classification) problem, set linout=T to
nn.fit <- nnet(medv/medv_max ~ ., data=BostonHousing, size=2, linout=TRUE, skip=TRUE)
```

```
## # weights:  44
## initial  value 74741861.304544
## iter  10 value 6690867.797615
## iter  20 value 192759.270122
## iter  30 value 249.659118
## iter  40 value 4.395220
## iter  50 value 4.392797
## iter  60 value 4.387729
## iter  70 value 4.387678
## iter  80 value 4.387377
## final  value 4.387374
## converged
```

```r
nn.predict <- predict(nn.fit)
# scale back the predictions
nn.predict = nn.predict*medv_max
```

Compute the training MSE and compare it to the MSE from part (b).

```r
mean((nn.predict - BostonHousing$medv)^2)
```

```
## [1] 21.67675
```

We find the following:

Linear regression MSE = 21.89483

NNET regression MSE = 21.67675

Using the same model you chose in part (c), fit a neural network. Compare the training MSEs between the two.

```r
medv_max = max(log(medv))
nn.fit <- nnet(log(medv)/medv_max~crim+chas+nox+rm+dis+rad+tax+ptratio+b+sqrt(lstat), data=BostonHousin
```

```
## # weights:  35
## initial  value 20967856.645275
## iter  10 value 1852930.119435
## iter  20 value 46804.102757
## iter  30 value 21574.863279
## iter  40 value 1.102733
## final  value 1.084192
## converged
```

```r
nn.predict <- predict(nn.fit)
# scale back the predictions
nn.predict = exp(nn.predict*medv_max)
mean((nn.predict - BostonHousing$medv)^2)
```

```
## [1] 16.75603
```

Linear regression MSE is 17.10186

NNET regression MSE = 16.75603

Finally submit BOTH your .rmd file and the resulting .pdf file with Canvas as Data Analysis Assignment 3.