# Notes of ISLR

Chapter 1: Linear Regression

*Varun Khanna*

*30 August 2018*

### Loading Libraries

```
LoadLibraries <-
  function(){
    library("MASS")
    library("ISLR")
    library("car")
    print("The libraries have been loaded")
  }
#loading the libraries
LoadLibraries()
```

```
## Loading required package: carData
## [1] "The libraries have been loaded"
```

### Reading in the Data

```
#Reading the Advertising Dataset available on www.statlearning.com
adv_data <- read.csv("Advertising.csv")

#Simple Linear Regression
#Loading the data from the Mass package
fix("Boston")
colnames(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

### Fitting in the basic LM model

```
#simple linear model with one predictor
lm.fit = lm(medv ~ lstat, data = Boston)

#we can also attach the column names and the run without the data param.
attach(Boston)
lm.fit <- lm(medv ~ lstat)

#detailed info about the model
summary(lm.fit)
```

```
##
```

```
## Call:
## lm(formula = medv ~ lstat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

### Plotting the model

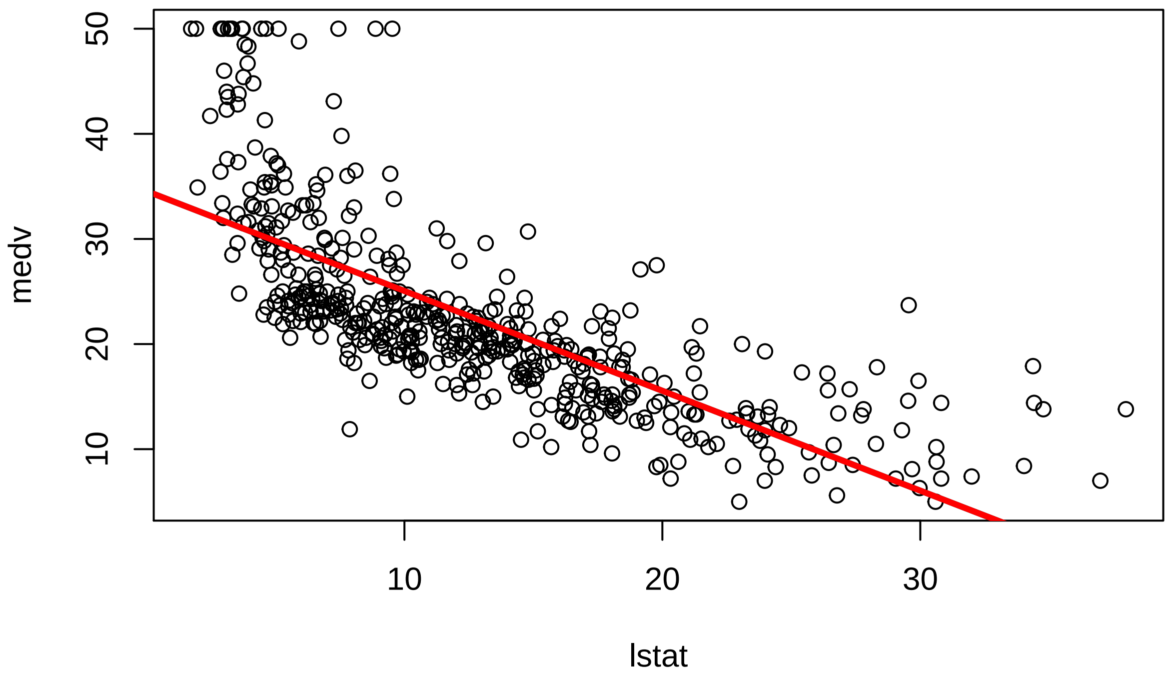
```r
#getting the confidence intervals for the coefficient estm.
confint(lm.fit)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

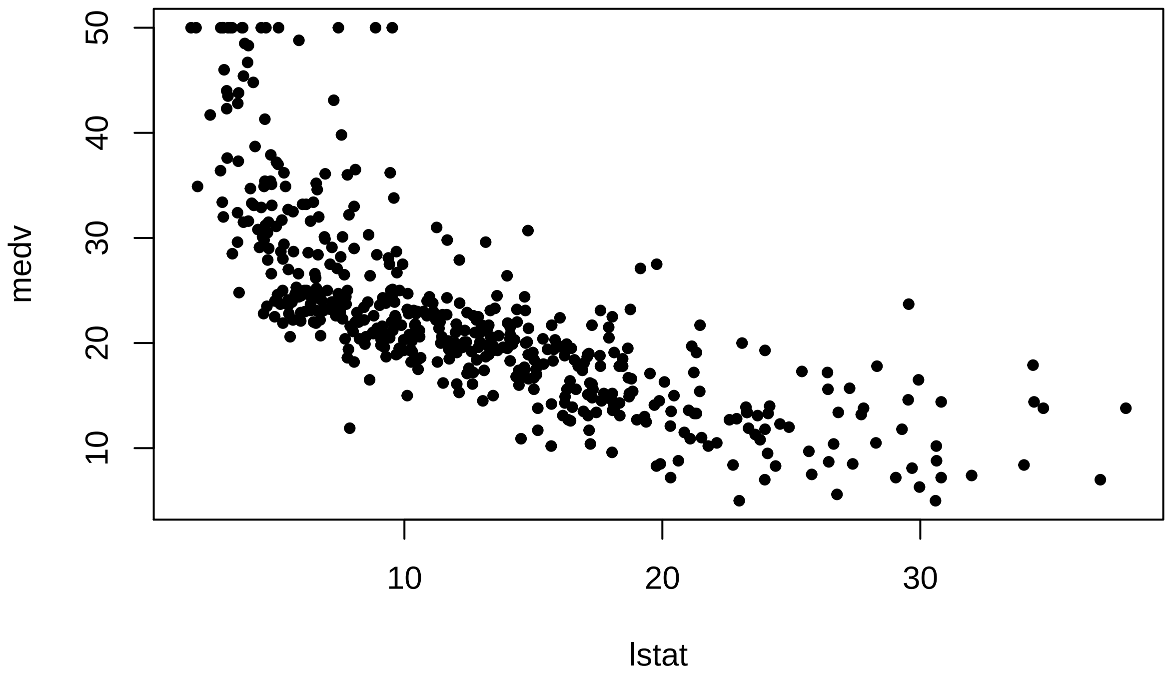
```r
#predict() can be used to produce CI and PI
#PI will be wider than the CI as the include the irreducible errors.
predict(lm.fit, data.frame (lstat= c(5,10,15)), interval = "prediction")
```

```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

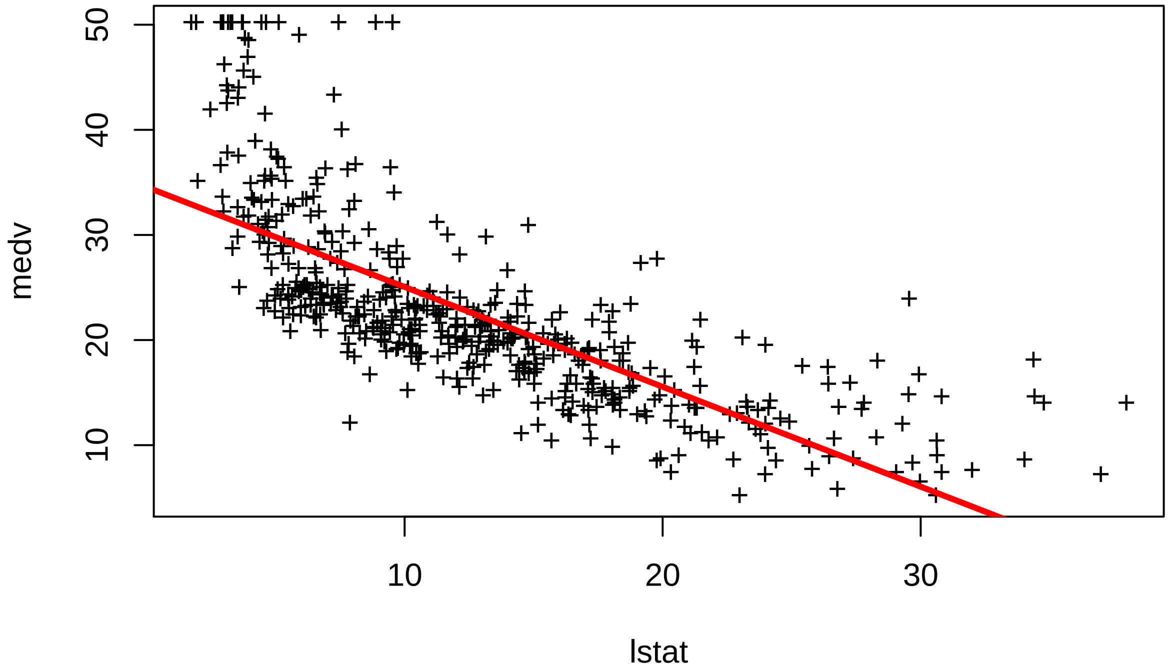
```r
#plotting the medv vs lstat (median inc. vs % of households in lower inc.)
plot(lstat, medv)
abline(lm.fit)

#There is some evidence for non-linearity in the relationship between lstat and medv.
#We will explore this issue later in this lab.

#For now, lets focus on improving the plot that we created above.
abline(lm.fit, lwd = 3)
abline(lm.fit, lwd = 3, col = "red")
```



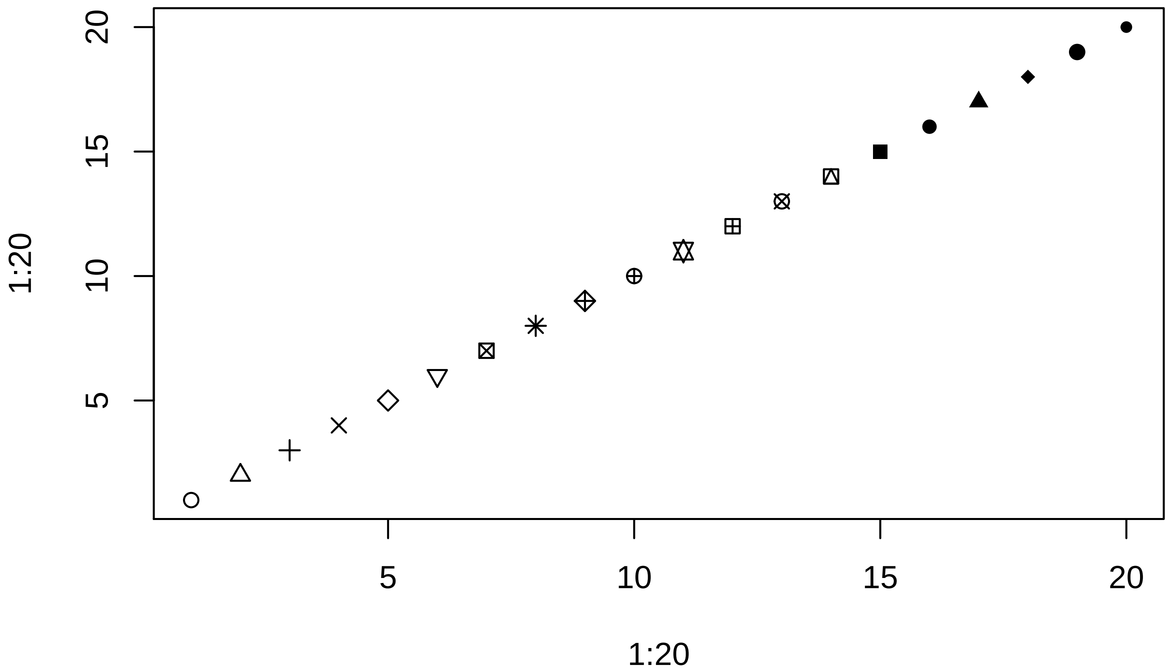

```r
plot(lstat, medv, pch = 20)
```




```r
plot(lstat, medv, pch = "+")
abline(lm.fit, lwd = 3, col = "red")
```

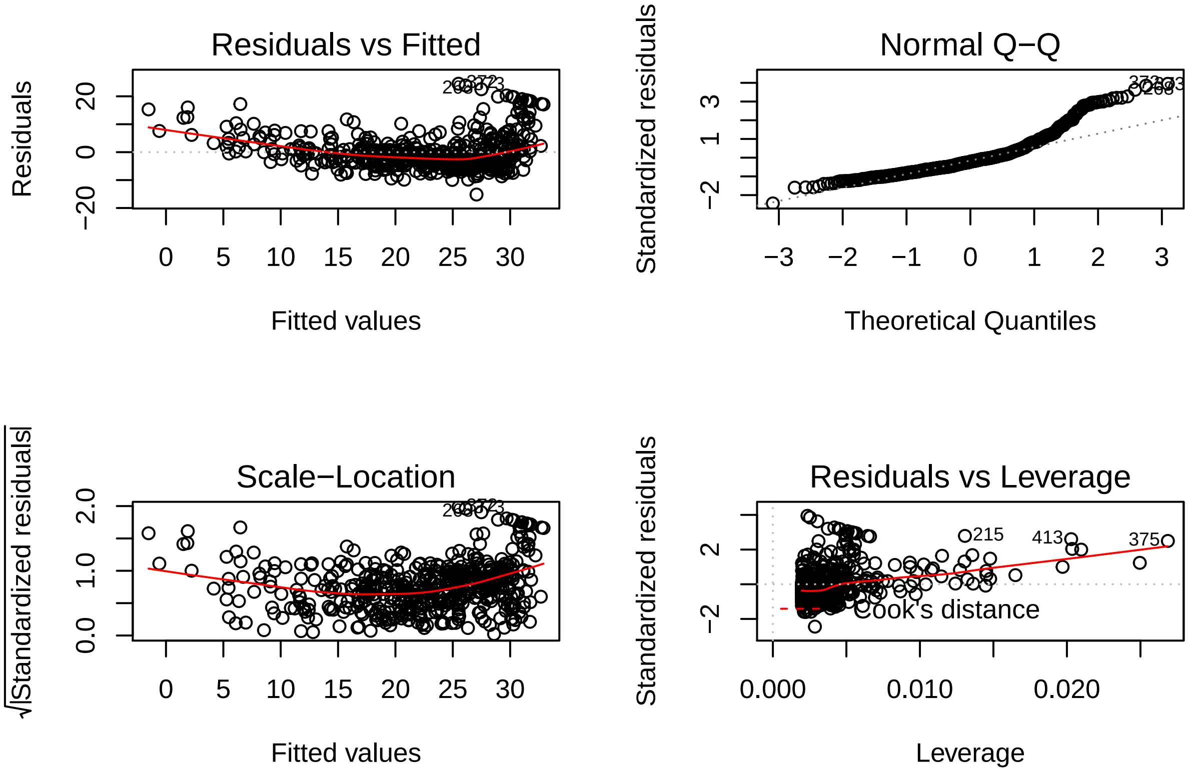



```
#all the symbols that are there in the plot func. through pch=""
plot(1:20, 1:20, pch = 1:20)
```




#### Plotting the 4 diagnostic plots

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




### Leverage statistics plots

These are used when there is some evidence of some non-linearity in the data as we can see in the above plots.

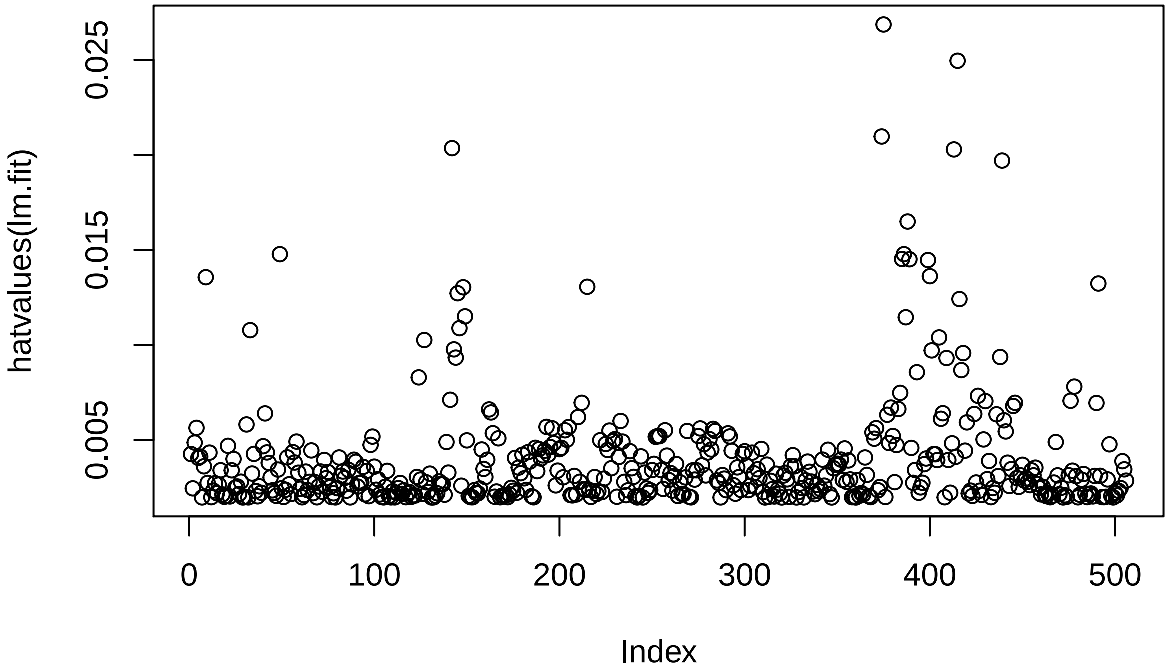
Leverage Statistics is defined as follows:

$$h_i = (\frac{1}{n}) + \frac{(x_i + \bar{x})^2}{\sum_i^n (x_i - \bar{x})^2}$$

It is clear that $h_i$ increases with $x_i$'s distance from its sample mean $\bar{x}$, thus the higher the leverage statistic for an observation, that point can be removed from the analysis. Ofcourse this becomes problematic when the data itself is non-linear and can be fit by a linear regression.

Finding out the highest leverage statistic is easy through the hatvalues function.

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




```
which.max(hatvalues(lm.fit))
```

```
## 375
## 375
```

Thus we can see that the 375'th observation has the highest Leverage statistics.

### Multiple Linear Regression

```
lm.fit = lm(medv~lstat + age, data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

We can even use all the predictors in the data set.

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

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## 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
## chas         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 ***
## black        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
```

WE can access all the components of the regression fit using the dollar operator on summary(). For example to access the fraction of the variance explained by the model, i.e. $R^2$ as below

```r
summary(lm.fit)$r.squared
```

```
## [1] 0.7406427
```

```r
#OR
summary(lm.fit)$r.sq
```

```
## [1] 0.7406427
```

We can also access the Variation Inflation Factor (VIF) to detect colinarity among the different predictors. Mathematically, it is defined as :

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}$$

A VIF of greater than 5 or 10 can be problematic. Some of the solutions to colinear variables are: 1. Dropouts 2. Average Standardized Versions, i.e. Coming two variables into one through some standardized averaging techniques. For eg.: Combine “limit” and “rating” to create “credit worthiness”.

```
#'car' lib has the function vif()

vif(lm.fit)
```

```
##     crim       zn    indus     chas      nox       rm      age      dis
## 1.792192 2.298758 3.991596 1.073995 4.393720 1.933744 3.100826 3.955945
##      rad      tax  ptratio    black    lstat
## 7.484496 9.008554 1.799084 1.348521 2.941491
```

From the above analysis, we can see that the variables age and indus have high p values and variables rad and tax seem to have higher than normal VIF. Let us just remove three variables age, indus and tax

```
lm.fit <- lm(medv ~ . -age - indus -tax -rad , data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ . - age - indus - tax - rad, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.803  -2.832  -0.625   1.454  27.766
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  29.507997   4.872538   6.056 2.76e-09 ***
## crim         -0.061174   0.030377  -2.014 0.044567 *
## zn            0.042032   0.013422   3.131 0.001842 **
## chas          3.029924   0.868349   3.489 0.000527 ***
## nox         -16.088513   3.232702  -4.977 8.93e-07 ***
## rm            4.149667   0.407685  10.179  < 2e-16 ***
## dis          -1.431665   0.188603  -7.591 1.59e-13 ***
## ptratio      -0.838640   0.117342  -7.147 3.19e-12 ***
## black         0.008292   0.002688   3.084 0.002153 **
## lstat        -0.525004   0.048351 -10.858  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.833 on 496 degrees of freedom
## Multiple R-squared:  0.7288, Adjusted R-squared:  0.7239
## F-statistic: 148.1 on 9 and 496 DF,  p-value: < 2.2e-16
```

We can see that F statistic has slightly increased, i.e. the current model is explained greater variance in the data.

### Interaction Terms in the model.

```
summary(lm(medv ~lstat*age, data= Boston))
```

```
##
```

```
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

### Non-linear Trasnformation of the predictors

In other words we would perform a polynomial regression. You can literally fit an 'n' degree polynomial to a given data and it would start fitting the data well. Alas, it just leads to overfitting.

```
lm.fit2 = lm(medv~lstat + I(lstat^2))
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

You can notice a significant jump in the F statistic.

We can use anova() to further quantify the extent to which the quadratic fit is superior to the linear fit.

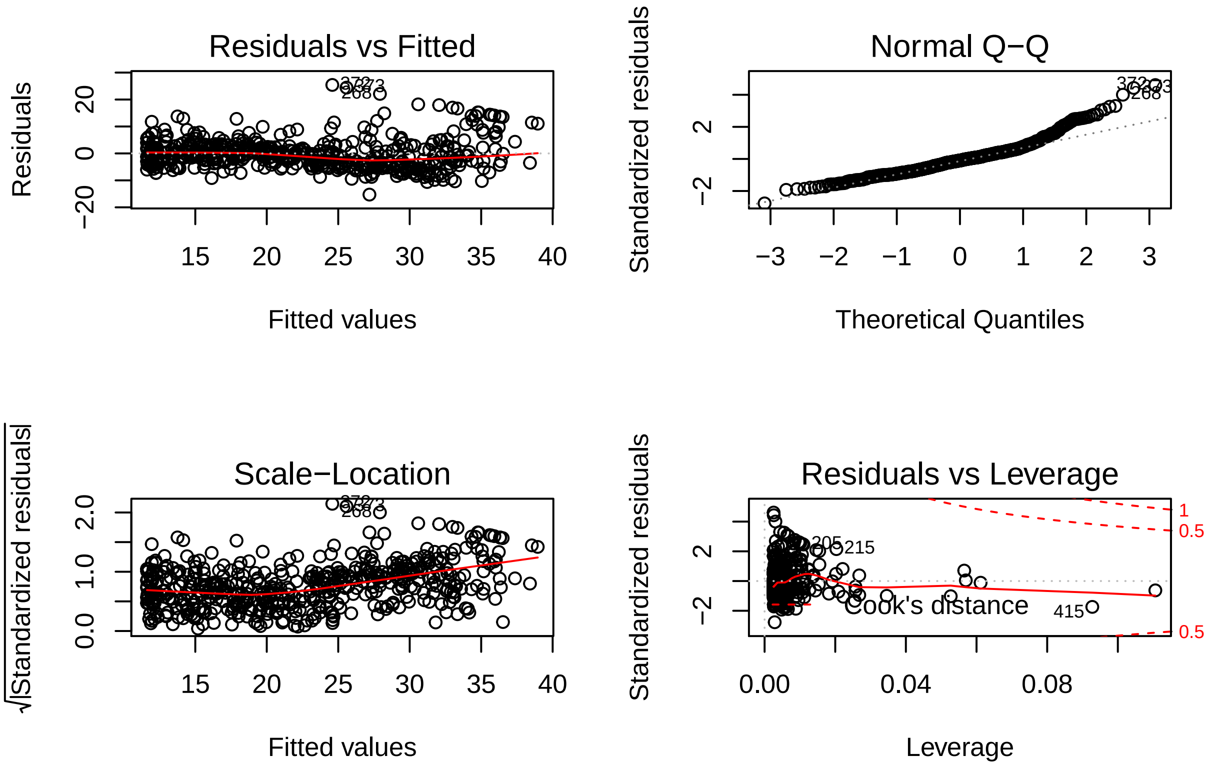
```r
lm.fit <- lm(medv~ lstat)
anova(lm.fit, lm.fit2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ lstat + I(lstat^2)
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    504 19472
## 2    503 15347  1    4125.1 135.2 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Direct Quote from the book :

"The anova() function performs a hypothesis test comparing the two models. The null hypothesis is that the two models fit the data equally well, and the alternative hypothesis is that the full model is superior. Here the F-statistic is 135 and the associated p-value is virtually zero. This provides very clear evidence that the model containing the predictors lstat and lstat2 is far superior to the model that only contains the predictor lstat. This is not surprising, since earlier we saw evidence for non-linearity in the relationship between medv and lstat"

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




As noted earlier, we can fit any degree polynomial to the data and we can do that using the poly() function inside lm()

```r
lm.fit6 <- lm(medv~ poly(lstat, 6))
summary(lm.fit6)
```

```
##
```

```
## Call:
## lm(formula = medv ~ poly(lstat, 6))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.7317  -3.1571  -0.6941   2.0756  26.8994
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         22.5328     0.2317  97.252  < 2e-16 ***
## poly(lstat, 6)1 -152.4595     5.2119 -29.252  < 2e-16 ***
## poly(lstat, 6)2   64.2272     5.2119  12.323  < 2e-16 ***
## poly(lstat, 6)3  -27.0511     5.2119  -5.190 3.06e-07 ***
## poly(lstat, 6)4   25.4517     5.2119   4.883 1.41e-06 ***
## poly(lstat, 6)5  -19.2524     5.2119  -3.694 0.000245 ***
## poly(lstat, 6)6    6.5088     5.2119   1.249 0.212313
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.212 on 499 degrees of freedom
## Multiple R-squared:  0.6827, Adjusted R-squared:  0.6789
## F-statistic: 178.9 on 6 and 499 DF,  p-value: < 2.2e-16
```

WE can even use other transformations like the log transformations and look at the model performance.

```r
summary(lm(medv~ log(rm), data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ log(rm), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.487  -2.875  -0.104   2.837  39.816
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -76.488      5.028  -15.21   <2e-16 ***
## log(rm)       54.055      2.739   19.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.915 on 504 degrees of freedom
## Multiple R-squared:  0.4358, Adjusted R-squared:  0.4347
## F-statistic: 389.3 on 1 and 504 DF,  p-value: < 2.2e-16
```

==========================================================

### Qualitative Predictors

We can load the Carseats dataset is part of the ISLR library.

It has a bunch of quant predictors but also has one qualitative predictor - “ShelveLoc” that records the shelf location of these carseats. It has three levels - Bad, good, medium. When we run qualitative variables in the lm(), R automatically creates dummy variables for these.

We can use contrasts() function to find the default for the ShelveLoc variable and can also tweak it to make our own.

```
contrasts(Carseats$ShelveLoc)
```

```
##        Good Medium
## Bad       0      0
## Good      1      0
## Medium    0      1
```

Lets run our regression model and look at the output of such a qualitative variable. We have also thrown some interaction terms into the model.

```
lm.fit = lm(Sales~ . +Income:Advertising+ Price:Age, data = Carseats)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Sales ~ . + Income:Advertising + Price:Age, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9208 -0.7503  0.0177  0.6754  3.3413
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        6.5755654  1.0087470   6.519 2.22e-10 ***
## CompPrice          0.0929371  0.0041183  22.567  < 2e-16 ***
## Income             0.0108940  0.0026044   4.183 3.57e-05 ***
## Advertising        0.0702462  0.0226091   3.107 0.002030 **
## Population         0.0001592  0.0003679   0.433 0.665330
## Price             -0.1008064  0.0074399 -13.549  < 2e-16 ***
## ShelveLocGood      4.8486762  0.1528378  31.724  < 2e-16 ***
## ShelveLocMedium    1.9532620  0.1257682  15.531  < 2e-16 ***
## Age               -0.0579466  0.0159506  -3.633 0.000318 ***
## Education         -0.0208525  0.0196131  -1.063 0.288361
## UrbanYes           0.1401597  0.1124019   1.247 0.213171
## USYes             -0.1575571  0.1489234  -1.058 0.290729
## Income:Advertising  0.0007510  0.0002784   2.698 0.007290 **
## Price:Age          0.0001068  0.0001333   0.801 0.423812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 386 degrees of freedom
## Multiple R-squared:  0.8761, Adjusted R-squared:  0.8719
## F-statistic:   210 on 13 and 386 DF,  p-value: < 2.2e-16
```