3.6.1 Libraries

The library() function is used to load libraries, or groups of functions and data sets that are not included in the base R distribution.

```
library(MASS)
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.3.3
```

3.6.2 Simple Linear Regression

The MASS library contains the Boston data set, which records medv for 506 neighborhoods around Boston. We will seek to predict medv using 13 predictors such as rm, age, and lstat.

```
#fix(Boston)
names(Boston)
```

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

We will start by using the lm() function to fit a simple linear regression model, with medv as the response and lstat as the predictor.

```
lm.fit <- lm(medv~lstat,data=Boston)
```

If we type lm.fit, some basic information about the model is output. For more detailed information, we use summary(lm.fit).

```
lm.fit
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## 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
```

We can use the names() function in order to find out what other pieces of information are stored in lm.fit. Although we can extract these quantities by name, it is safer to use the extractor functions like coef() to access them.

```
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
coef(lm.fit)
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

In order to obtain a confidence interval for the coefficient estimates, we can use the confint() command

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

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

The predict() function can be used to produce confidence intervals and prediction intervals for the prediction of medv for a given value of lstat.

```
predict(lm.fit, data.frame(lstat=c(5,10,15)), interval = "confidence")
```

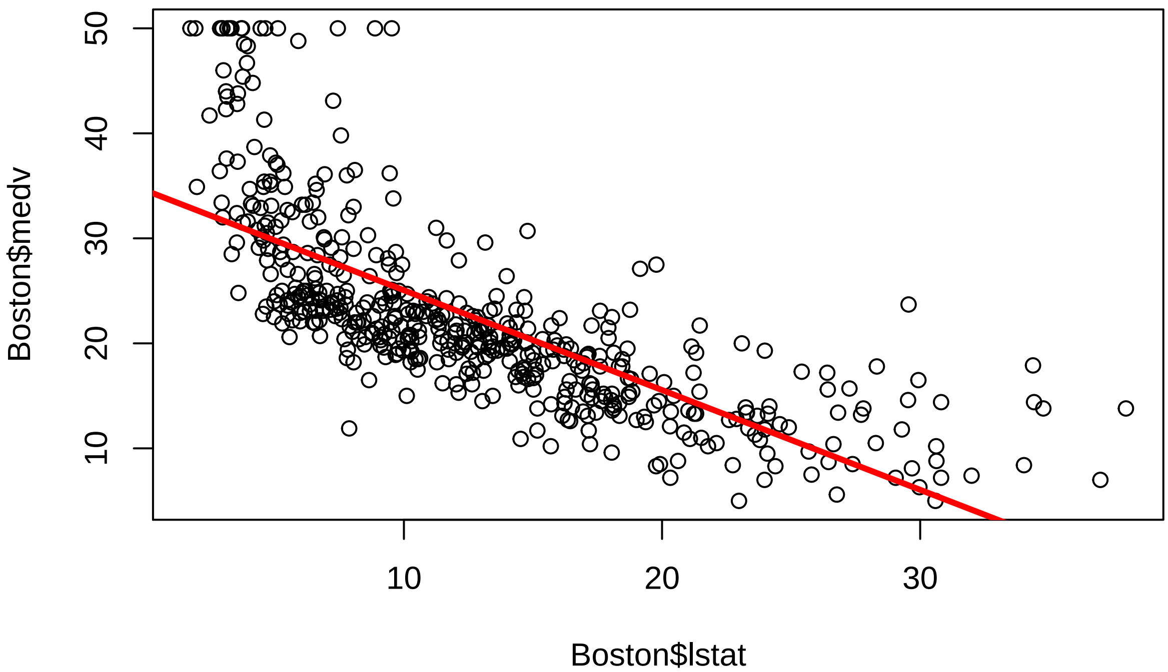
```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

```
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
```

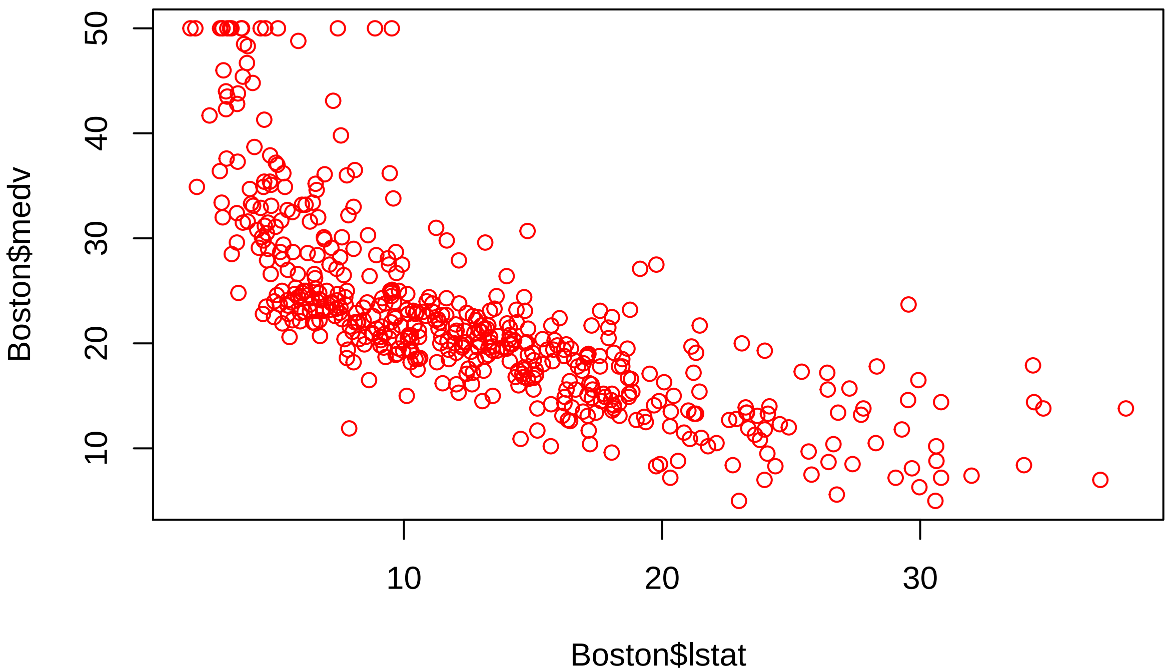
We will now plot medv and lstat along with the least squares regression line using the plot() and abline() functions

```
plot(Boston$lstat,Boston$medv)
abline(lm.fit, lwd=3, col="red")
```

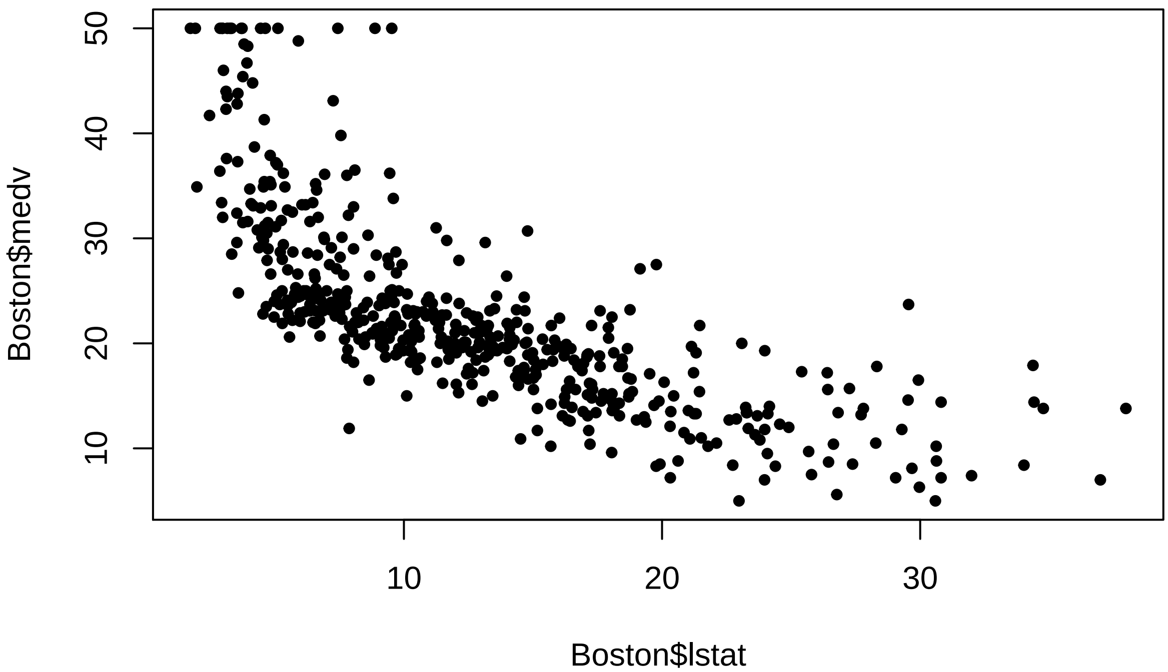


Below we experiment with some additional settings for plotting lines and points.

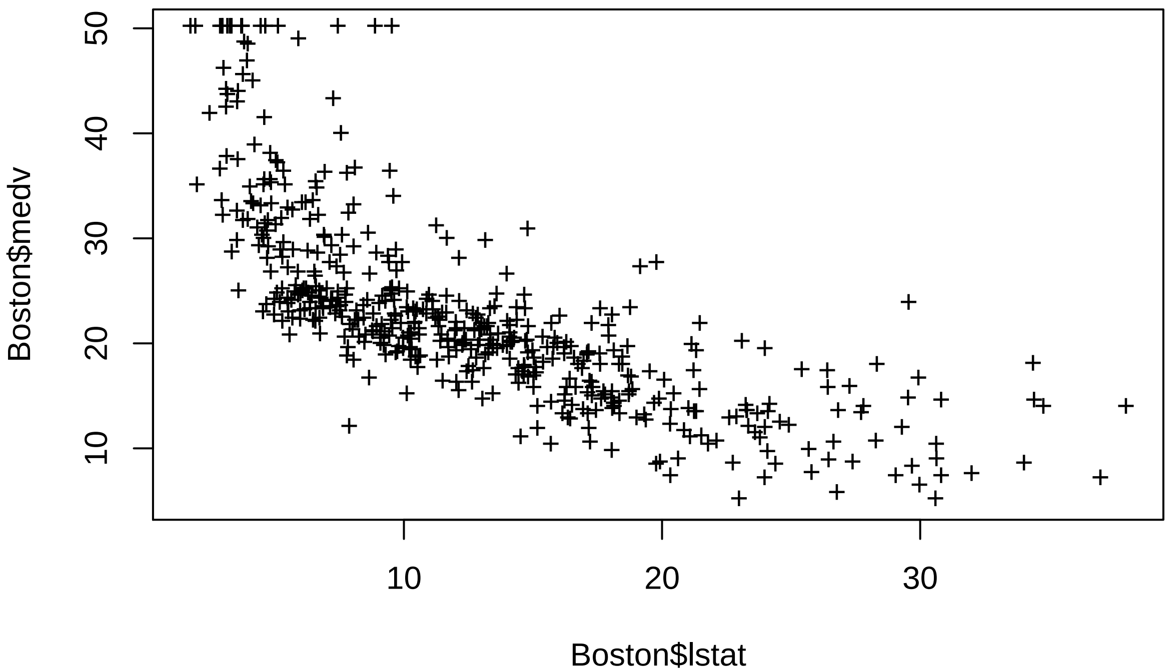
```
plot(Boston$lstat,Boston$medv,col="red")
```



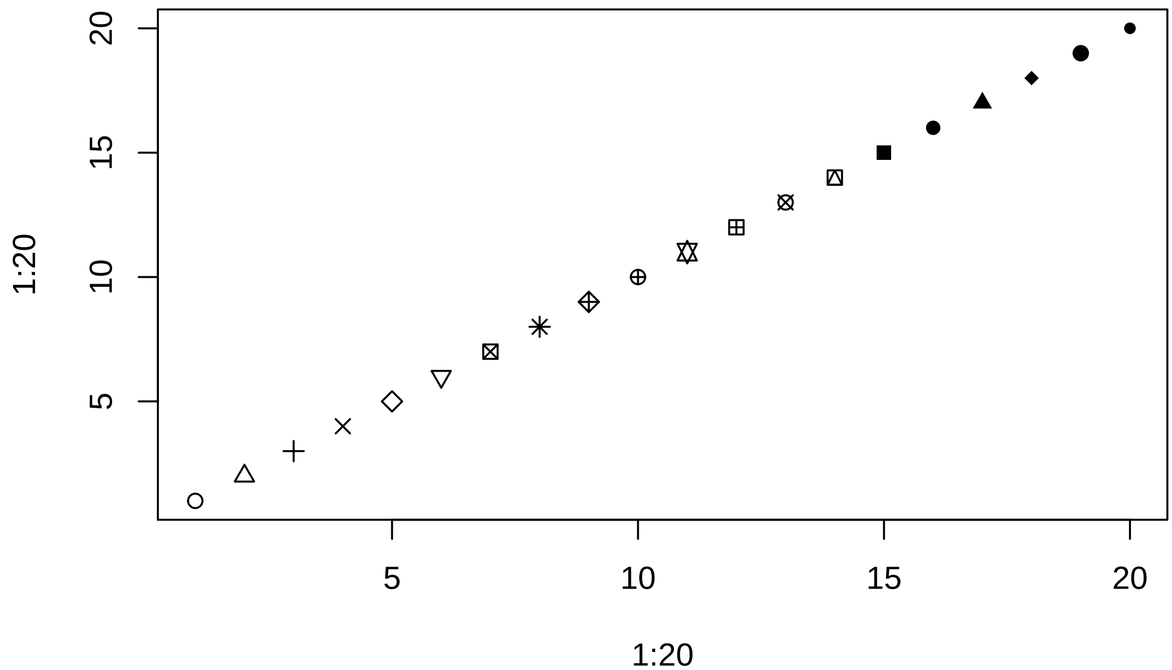

```
plot(Boston$lstat,Boston$medv,pch=20)
```




```
plot(Boston$lstat,Boston$medv,pch="+")
```

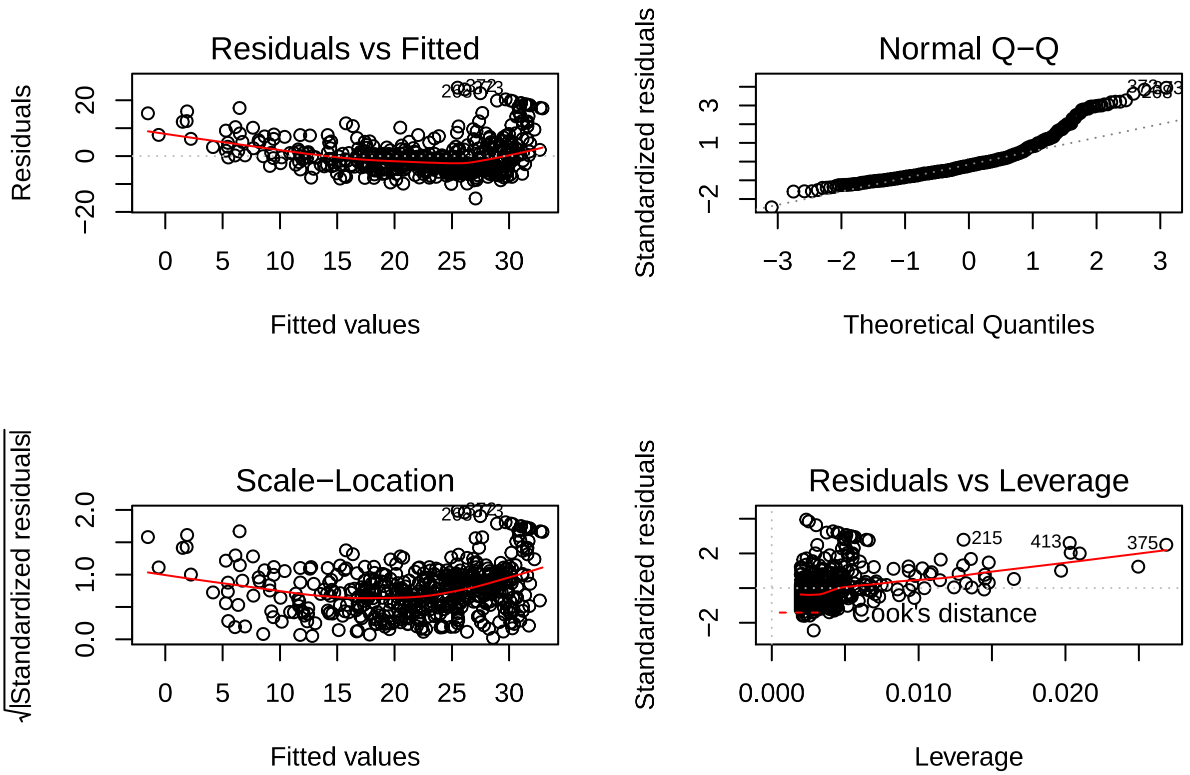



```
plot(1:20,1:20,pch=1:20)
```



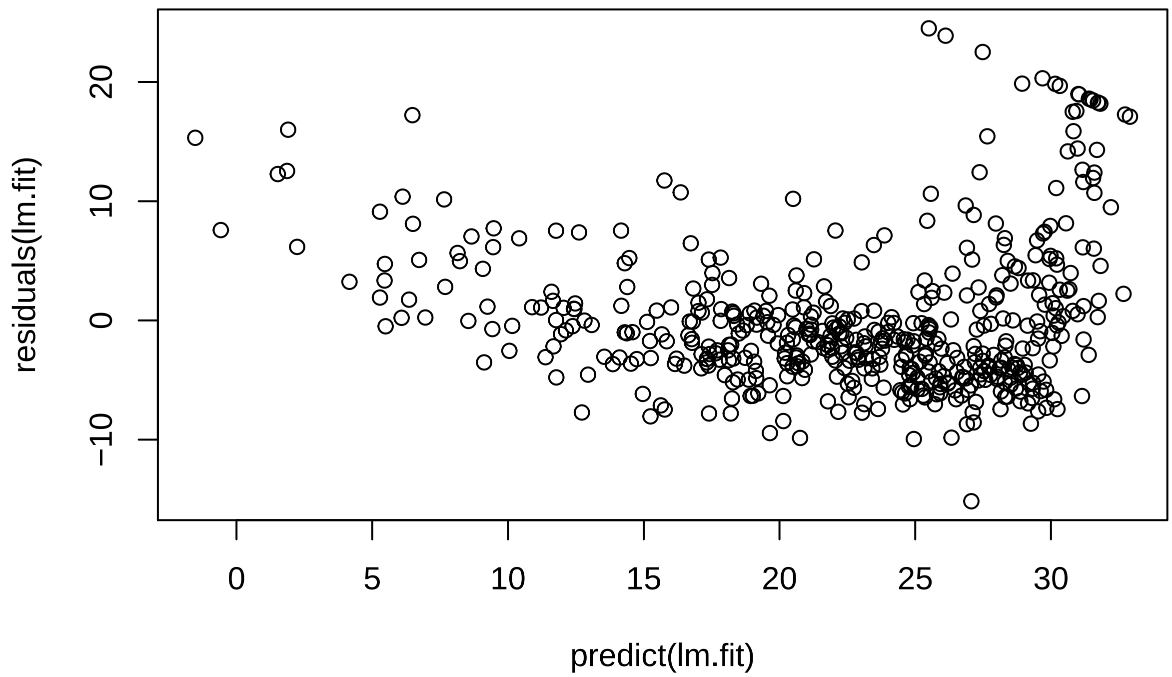
Next we examine some diagnostic plots. Four diagnostic plots are automatically produced by applying the plot() function directly to the output from lm(). In general, this command will produce one plot at a time, and hitting Enter will generate the next plot.

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

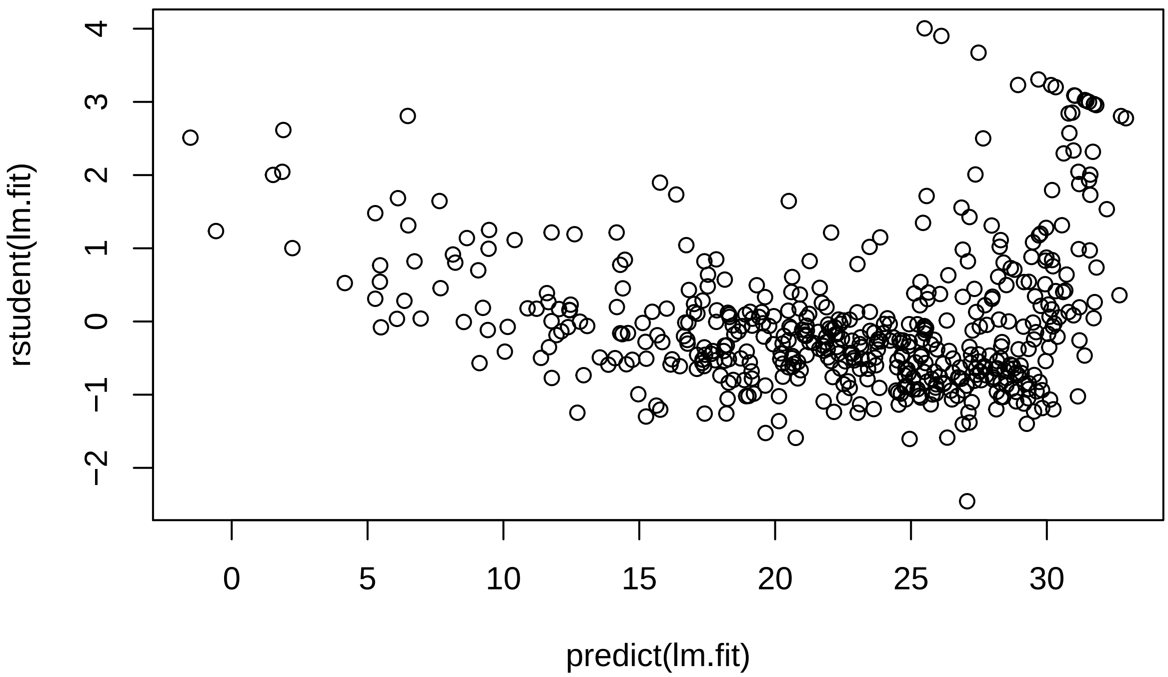


Alternatively, we can compute the residuals from a linear regression fit using the residuals() function. The function rstudent() will return the studentized residuals.

```
plot(predict(lm.fit), residuals(lm.fit))
```

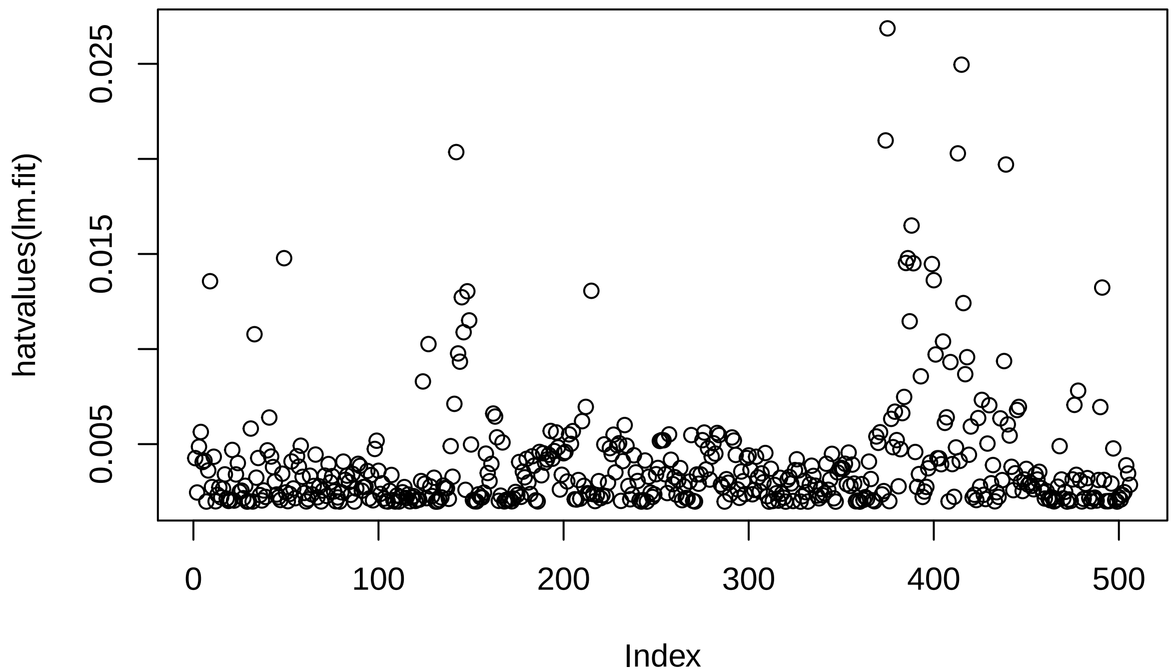



```r
plot(predict(lm.fit), rstudent(lm.fit))
```



On the basis of the residual plots, there is some evidence of non-linearity. Leverage statistics can be computed for any number of predictors using the hatvalues() function.

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



3.6.3 Multiple Linear Regression

In order to fit a multiple linear regression model using least squares, we again use the lm() function. The syntax lm(y~x1+x2+x3) is used to fit a model with three predictors.

```
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
```

The Boston data set contains 13 variables, and so it would be cumbersome to have to type all of these in order to perform a regression using all of the predictors.

```
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
```

The vif() function, part of the car package, can be used to compute variance inflation factors.

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.3.3
```

```
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
```

what if we would like to perform a regression using all of the variables but one?

```
lm.fit1 <- lm(medv~.-age,data=Boston)
summary(lm.fit1)
```

```
## 
```

```
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

Alternatively, the update() function can be used.

```
lm.fit1 <- update(lm.fit,~.-age)
```

3.6.4 Interaction Terms

It is easy to include interaction terms in a linear model using the lm() function. The syntax lstat:black tells R to include an interaction term between lstat and black. The syntax lstat*age simultaneously includes lstat, age, and the interaction term lstatxage as predictors.

```
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
```

3.6.5 Non-linear Transformations of the Predictors

The lm() function can also accomodate non-linear transformations of the predictors. For instance, given a predictor X, we can create a predictor X^2 using I(X^2)

```r
lm.fit2 <- lm(medv~lstat+I(lstat^2), data=Boston)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
##
## 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
```

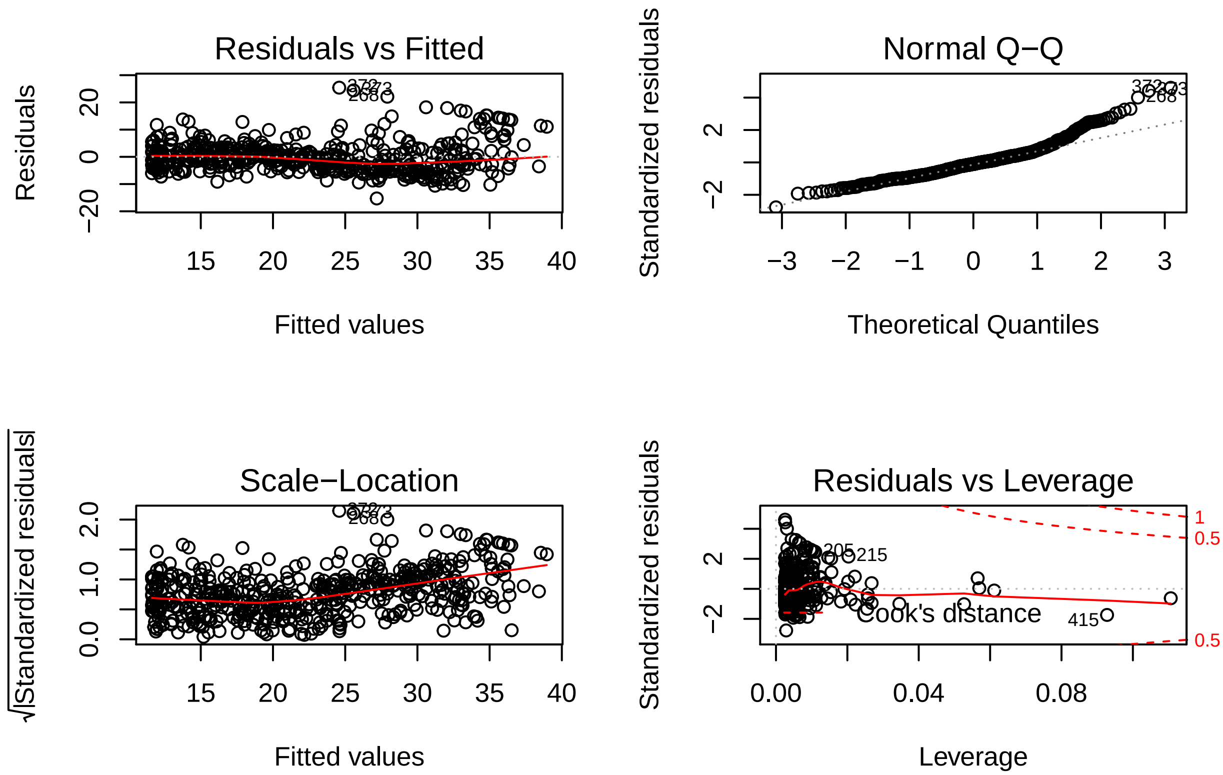
We use the anova() function to further quantify the extent to which the quadratic fit is superior to the linear fit.

```r
lm.fit <- lm(medv~lstat,data=Boston)
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
```

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.

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



In order to create a cubic fit, we can include a predictor of the form I(X^3). However, this approach can start to get cumbersome for higher order polynomials. A better approach involves using the poly() function to create the polynomial within lm().

```
lm.fit5 <- lm(medv~poly(lstat,5),data=Boston)
summary(lm.fit5)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 5), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5433  -3.1039  -0.7052   2.0844  27.1153
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.5328     0.2318  97.197  < 2e-16 ***
## poly(lstat, 5)1 -152.4595     5.2148 -29.236  < 2e-16 ***
## poly(lstat, 5)2   64.2272     5.2148  12.316  < 2e-16 ***
## poly(lstat, 5)3  -27.0511     5.2148  -5.187 3.10e-07 ***
## poly(lstat, 5)4   25.4517     5.2148   4.881 1.42e-06 ***
## poly(lstat, 5)5  -19.2524     5.2148  -3.692 0.000247 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.215 on 500 degrees of freedom
```

```
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6785
## F-statistic: 214.2 on 5 and 500 DF,  p-value: < 2.2e-16
```

Of course, we are in no way restricted to using polynomial transformations of the predictors. Here we try a log transformation.

```
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
```

3.6.6 Qualitative Predictors

We will attempt to predict Sales in 400 locations based on a number of predictors.

```
#fix(Carseats)
names(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

The predictor Shelveloc takes on three possible values, Bad, Medium, and Good. Given a qualitative variable such as Shelveloc, R generates dummy variables automatically.

```
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
```

The contrasts() function returns the coding that R uses for the dummy variables.

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

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

3.6.7 Wrting Functions

Below we provide a simple function that reads in the ISLR and MASS libraries, called LoadLibraries().

```
LoadLibraries=function(){
  library(ISLR)
  library(MASS)
  print("The libraries have been loaded.")
}
```

If we call the function, the libraries are loaded in and the print statement is output.

```
LoadLibraries()
```

```
## [1] "The libraries have been loaded."
```