Assignment 2

Conner McCloney

1

a)

This dataset is a collection of books that can be found on Goodreads, which is a company considers itself 'the world's largest site for readers and book recommendations', that allows users to browse its database of books, rate them, interact with other users on their platform, and other related activites. This dataset records the title of the book, its author(s), ISBN numbers, number of pages, language it's written in, average rating, number of ratings, and number of text reviews given.

The research question I used last week was 'Can we predict the average rating of a book from Goodreads?' In other words, I'm curious if any of these variables, besides the title of the book, can be used as predictors to accurately model the average rating of a book as a response variable.

The plots below model a few of these variables against the average rating, and all of them seem to have the same distributional shape, centering around an average rating of 4. If the value of the predictor is low, there seems to be no relationship between it and the average rating, but as you increase the value, the average rating will become more and more likely to be 4. I find it surprising that this relationship also exists between average rating and the number of pages in a book as well, which is something I didn't expect to see, and that according to the population of Goodreads users, it seems the more attention a book receives, as measured by the number of reviews and ratings given, the more the average rating will converge to a ~4/5 rating, which is interesting.

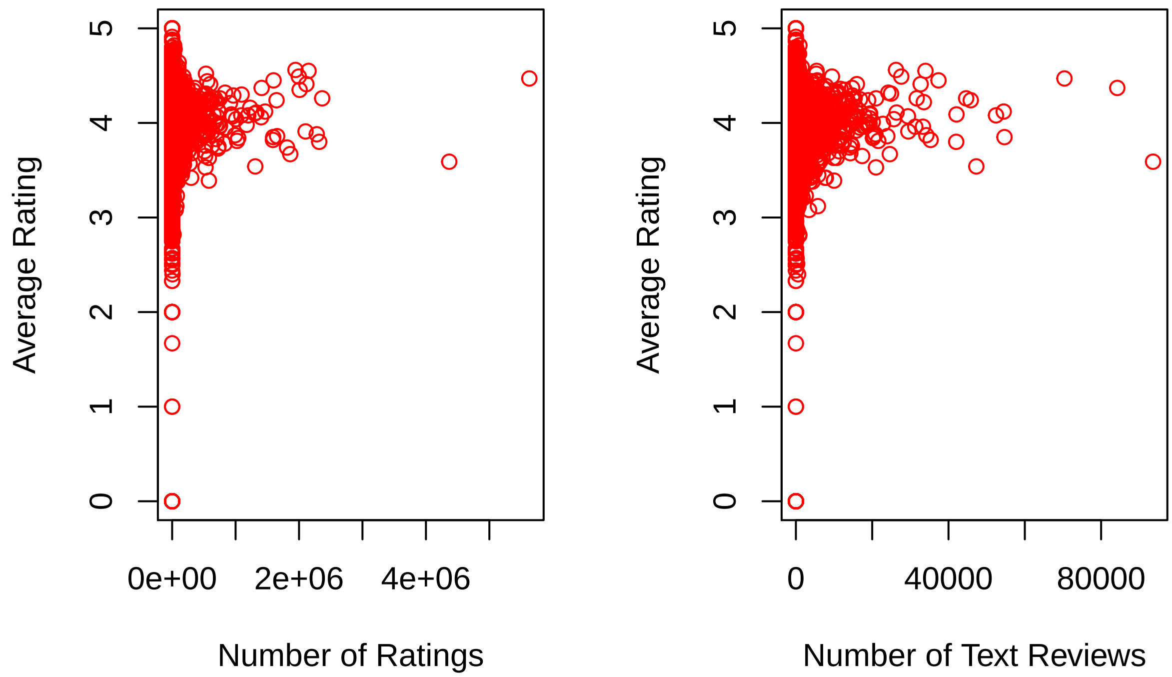
Source: [https://www.kaggle.com/jealousleopard/goodreadsbooks]

```
books_data <- read.csv("books.csv")
par(mfrow=c(1,2))
plot(as.numeric(as.character(average_rating))~as.numeric(as.character(ratings_count)),data=books_data,
```

```
## Warning in eval(predvars, data, env): NAs introduced by coercion
```

```
plot(as.numeric(as.character(average_rating))~as.numeric(as.character(text_reviews_count)),data=books_da
```

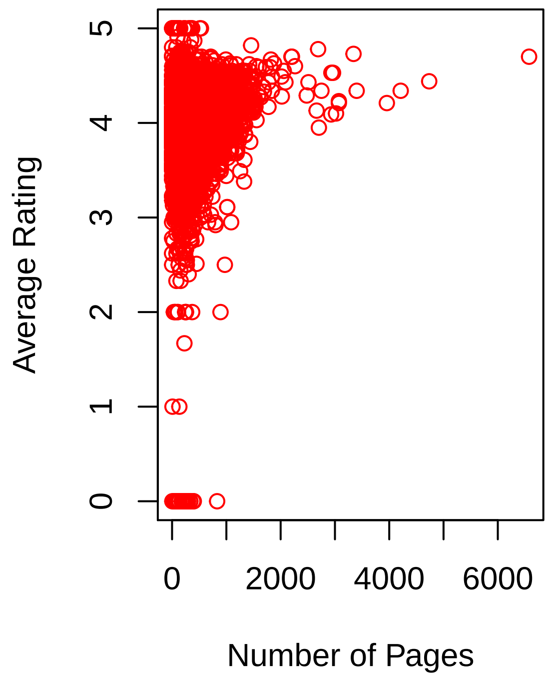
```
## Warning in eval(predvars, data, env): NAs introduced by coercion
```




```
plot(as.numeric(as.character(average_rating))~as.numeric(as.character(X..num_pages)),data=books_data, yl
```

```
## Warning in eval(predvars, data, env): NAs introduced by coercion

## Warning in eval(predvars, data, env): NAs introduced by coercion
```



b)

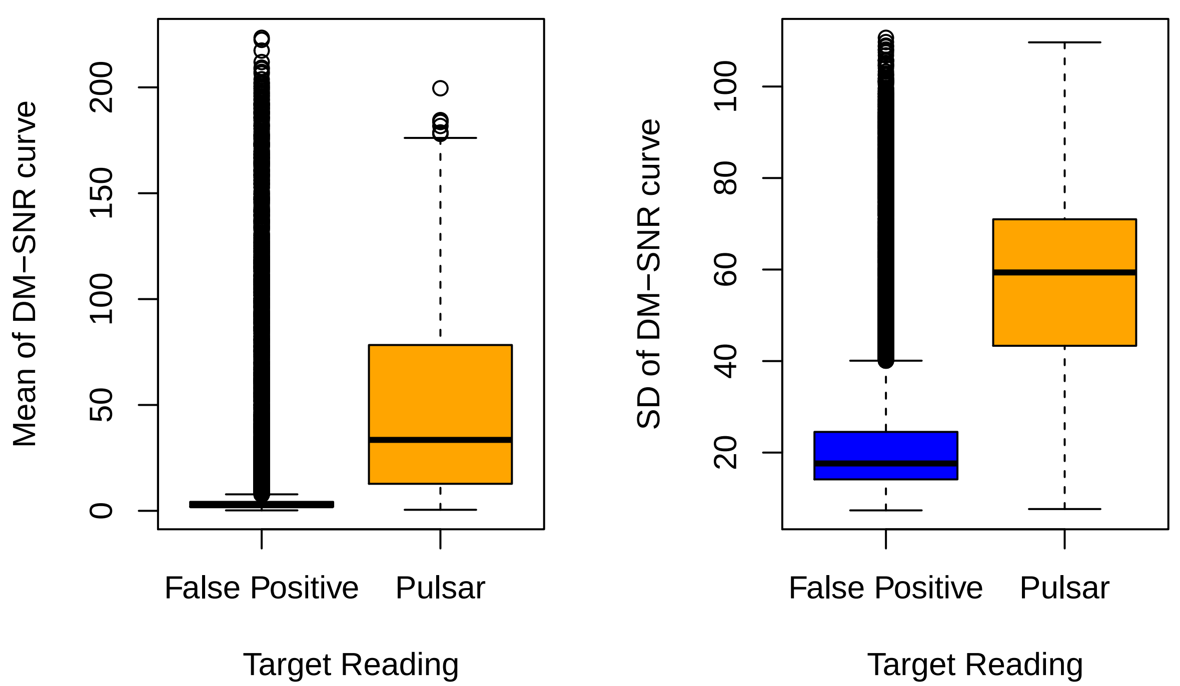
This dataset is a sample of pulsar candidates measured during the High Time Resolution Universe Survey in 2011. A pulsar is a rare type of rotating Neutron star that emits electromagnetic radiation, and can only be measured when this beam is aimed directly at Earth. These pulsars were searched for using large radio telescopes looking for periodic radio signals that a pulsar would produce. Many measurements taken that could be a pulsar, known as a candidate, are recorded, however in practice most observations recorded are caused by radio frequency interference (RFI) and noise. This dataset records the mean, standard deviation, skewness, and excess kurtosis of the integrated profile, which is an array of variables that describe the signal recorded, and of the DM-SNR curve. The DM-SNR curve, which stands for Dispersion Measure - Signal-to-Noise Ratio, describes the relationship between the two for the observed signal, where a curve whose SNR peaks at a DM of zero is likely RFI, or if it is a legitimate signal it should peak at a DM greater than zero. Lastly, there is the target_class variable, which is a binary classification variable of whether the given signal was truly a pulsar or not.

The research question I used last week was 'Is there a relationship between the dispersion measure signal-to-noise ratio (DM-SNR) curve and the classification of an observed measurement as a pulsar or a false positive?', which the plots below were used to explore, although extending this question to include the integrated profile would be equally interesting.

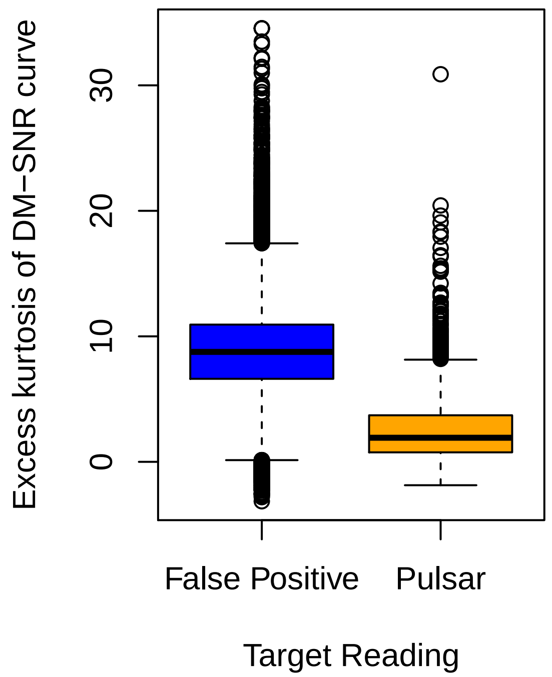
The plots below model each of the characteristics of the DM-SNR curve against the classification of whether a reading is a pulsar or a false positive. It seems fairly clear from the first two plots that the mean and SD of the curve tend to be larger if the signal is a legitimate pulsar signal, although the false positive variation, especially for the mean of the curve, are much smaller, resulting in a high number of outliers according to these plots. For the remaining two plots, the reverse seems to be true, as the excess kurtosis and skewness of the curve tend to be larger if the signal is a false positive. The variation in the kurtosis plot seems to be roughly equal for the two readings, although it is very small for true pulsars in the skewness plot.

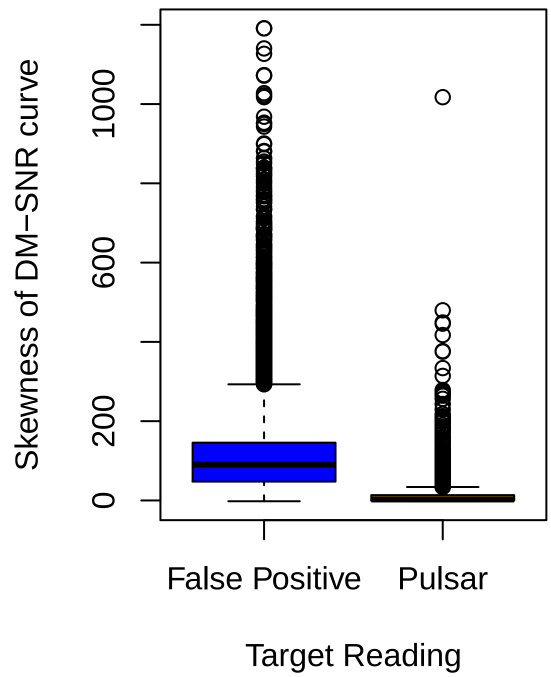
Source: [https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star]

```
pulsar_data <- read.csv("pulsar_stars.csv")
pulsar_data$target_class[pulsar_data$target_class==1] <- "Pulsar"
pulsar_data$target_class[pulsar_data$target_class==0] <- "False Positive"
par(mfrow=c(1,2))
boxplot(Mean.of.the.DM.SNR.curve~target_class,data=pulsar_data,ylab="Mean of DM-SNR curve", xlab="Targe
boxplot(Standard.deviation.of.the.DM.SNR.curve~target_class,data=pulsar_data, ylab="SD of DM-SNR curve"
```




```
par(mfrow=c(1,2))
boxplot(Excess.kurtosis.of.the.DM.SNR.curve~target_class,data=pulsar_data, ylab="Excess kurtosis of DM-S
boxplot(Skewness.of.the.DM.SNR.curve~target_class,data=pulsar_data, ylab="Skewness of DM-SNR curve", xla
```





3

The KNN classifier examines the K-nearest neaighbors of a given point, and classifies the response value of it based on the mode of those K-nearest neighbors. The KNN regression model examines the K-nearest neighbors of a given point x_0, and generates an estimate for $f(x_0)$ based on the mean of the response values of those points. This will generate $\hat{f}(X)$, in an attempt to find the true relationship between the response and the predictor variables.

4

```
#Simple Linear Regression
library(MASS)
library(ISLR)
##fix(Boston)
names(Boston)
```

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

```
##lm.fit = lm(medv~lstat)
attach(Boston)
lm.fit = lm(medv~lstat)
lm.fit
```

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

```r
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
```

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

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

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

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

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

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

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

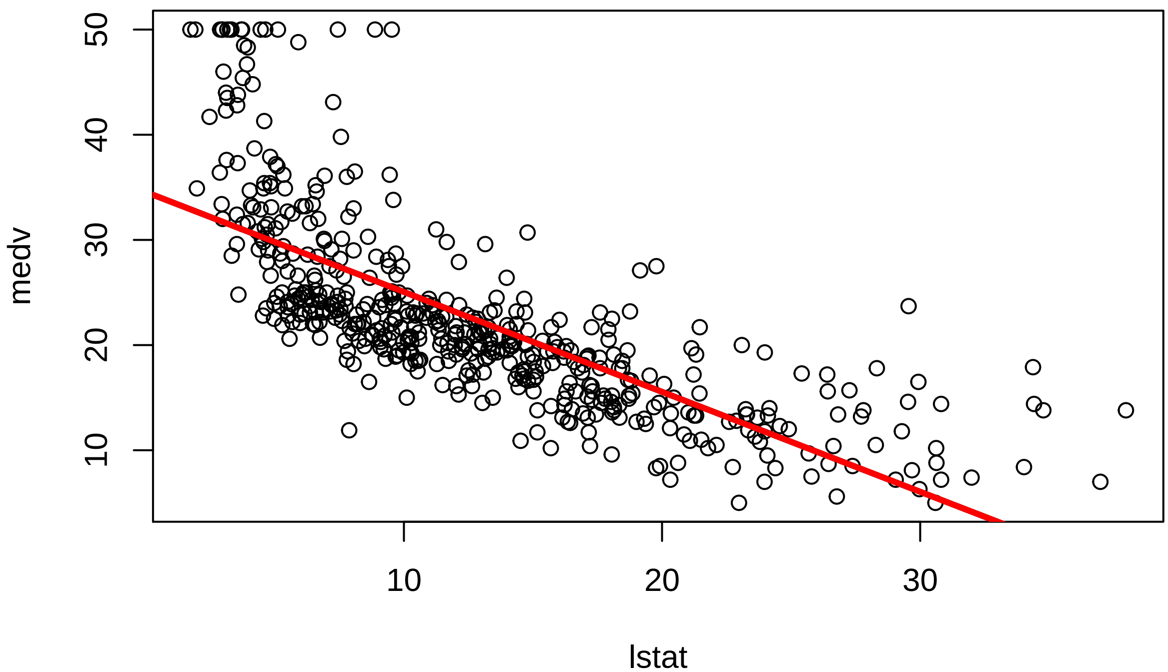
```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

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

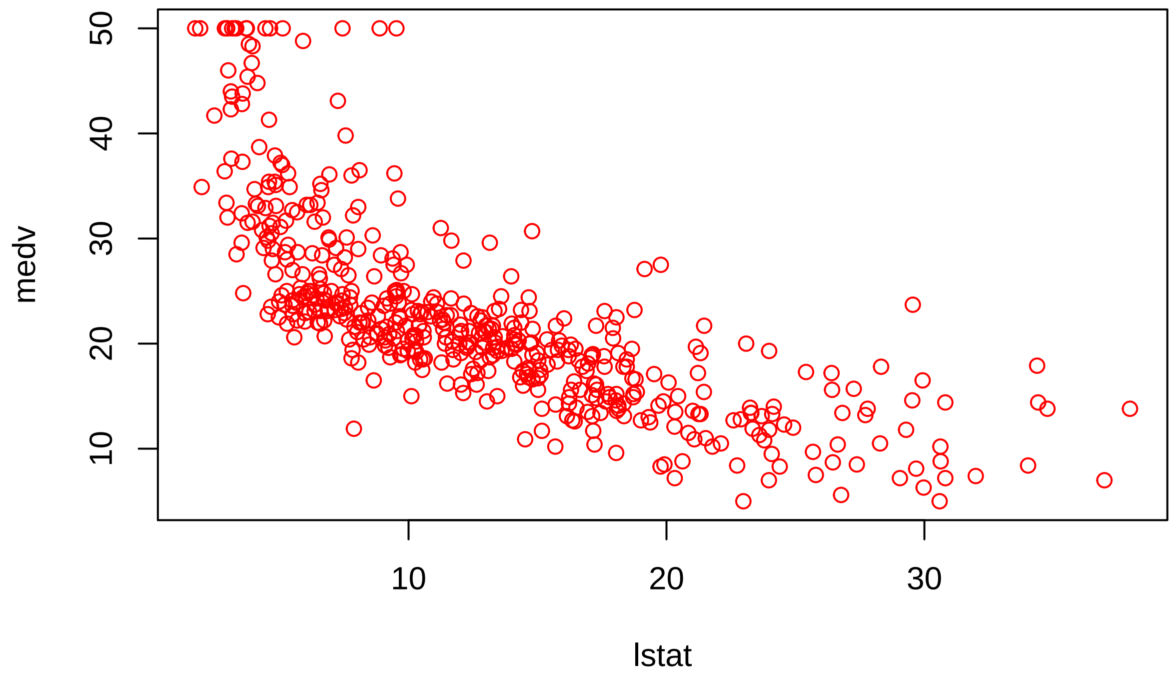
```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
```

```
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

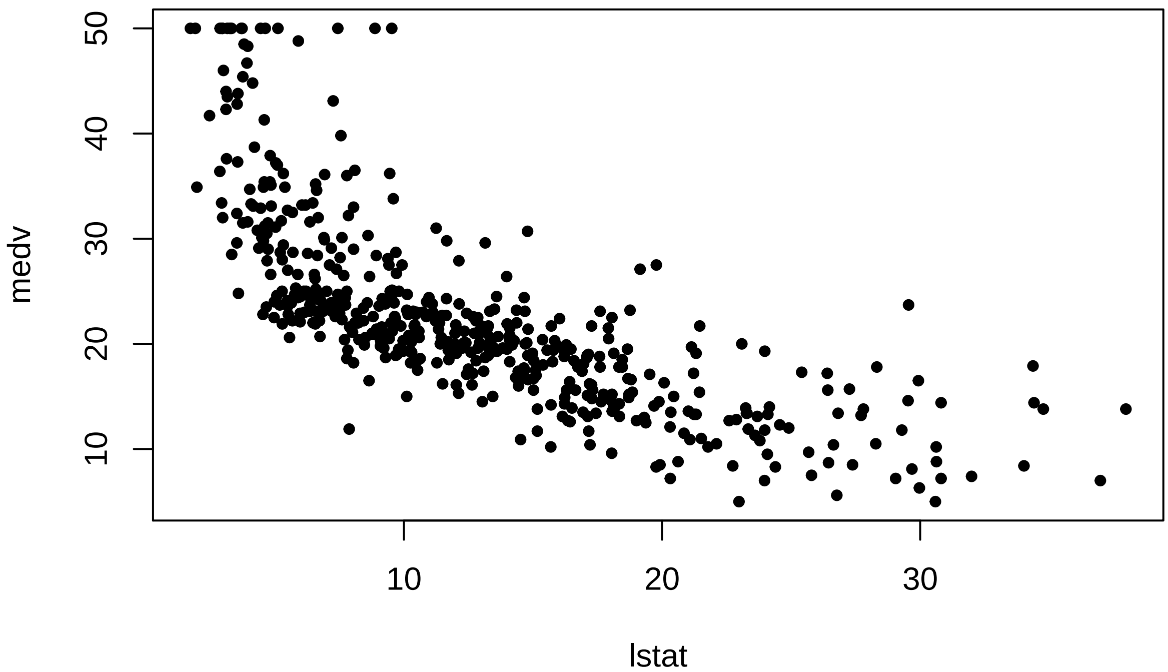
```
plot(lstat,medv)
abline(lm.fit)
abline(lm.fit,lwd=3)
abline(lm.fit,lwd=3,col="red")
```



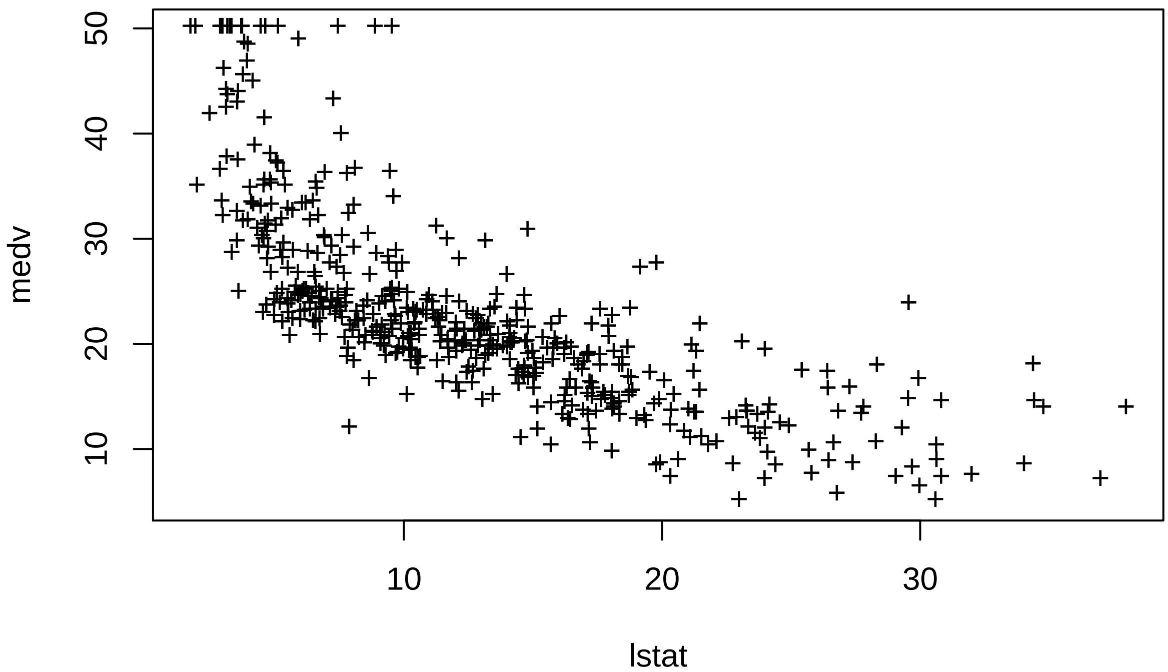

```
plot(lstat,medv,col="red")
```



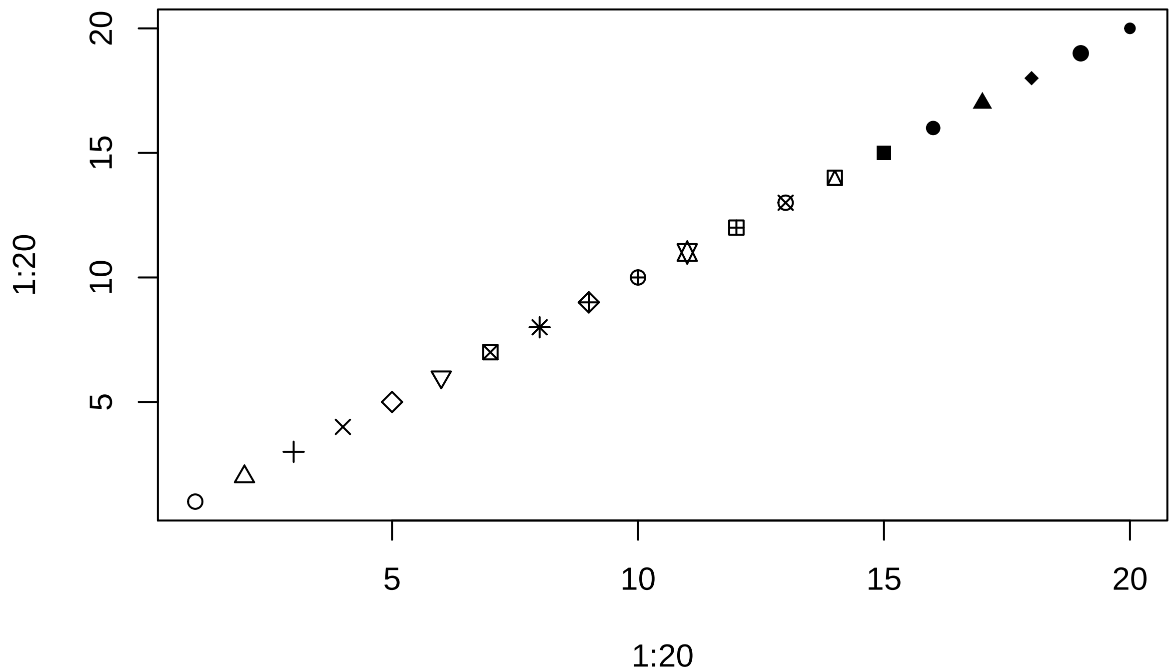

```
plot(lstat,medv,pch=20)
```



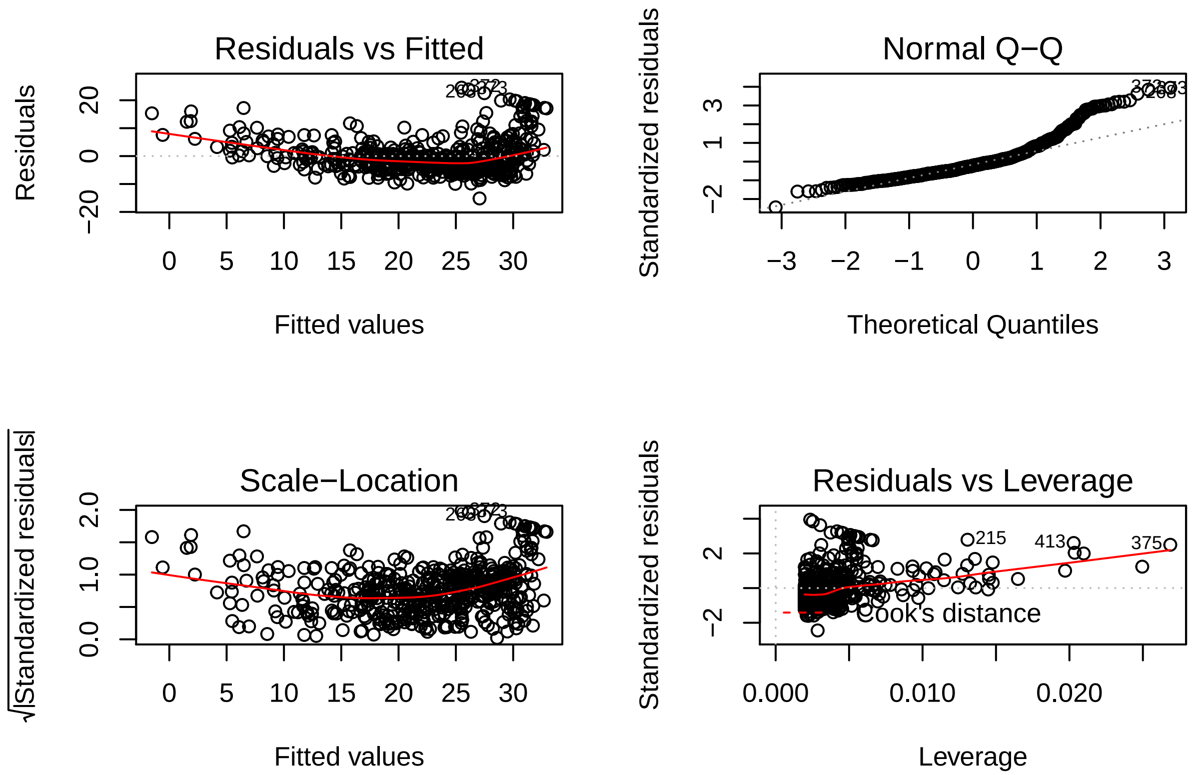

```r
plot(lstat,medv,pch="+")
```




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




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




```r
plot(predict(lm.fit), residuals(lm.fit))
plot(predict(lm.fit), rstudent(lm.fit))
plot(hatvalues(lm.fit))
which.max(hatvalues(lm.fit))
```

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

```r
#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
```

```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
```

```r
library(car)
```

```
## Loading required package: carData
```

```r
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
```

```r
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
```

```
lm.fit1=update(lm.fit,~.-age)
#Interaction Terms
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
```

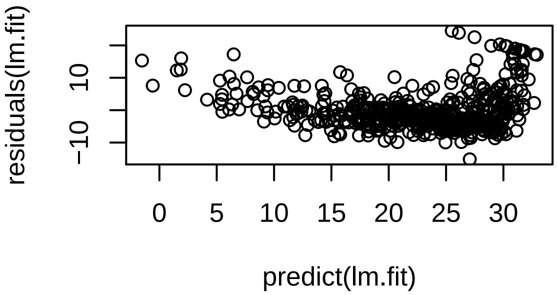
```
#Non-linear transformations of the Predictors
lm.fit2=lm(medv~lstat+I(lstat^2))
summary(lm.fit2)
```

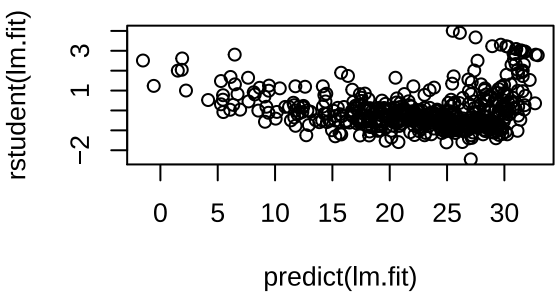
```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## 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
```

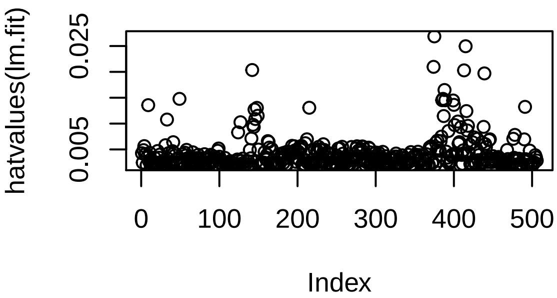
```r
lm.fit=lm(medv~lstat)
anova(lm.fit,lm.fit2)
```

```
## 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
```

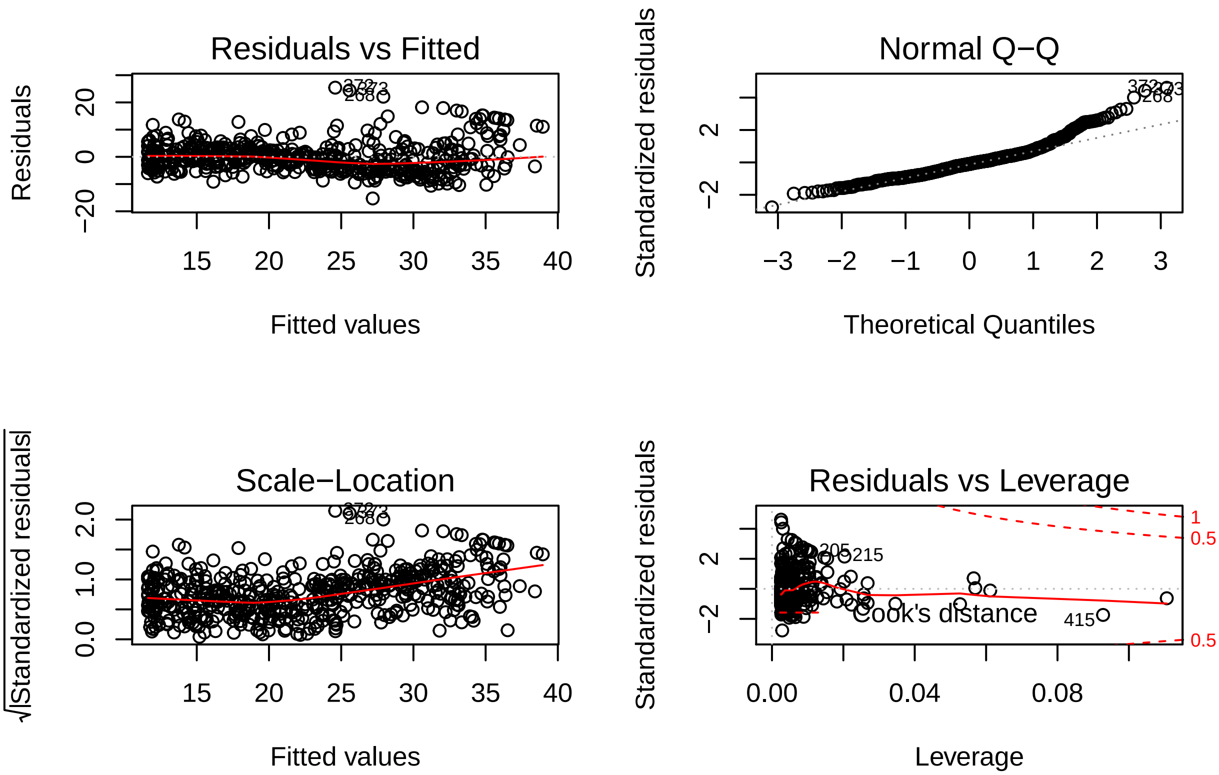
```r
par(mfrow=c(2,2))
```








```r
plot(lm.fit2)
```




```r
lm.fit5=lm(medv~poly(lstat,5))
summary(lm.fit5)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 5))
##
## 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
```

```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
```

```r
#Qualitative Predictors
##fix(Carseats)
names(Carseats)
```

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

```r
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
```

```r
attach(Carseats)
contrasts(ShelveLoc)
```

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

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

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

```r
LoadLibraries()
```

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

```r
#Export data for use in Python
write.csv(Boston,"Boston.csv")
write.csv(Carseats,"Carseats.csv")
```