Multiple Linear Regression

Dr Liwan Liyanage - Western Sydney University

2 July 2018

Multiple Linear Regression

First upload the data file

```r
library(MASS)
attach(Boston)
```

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, X1, X2, and X3.

```
lm.fit2 = lm(formula = medv ~ lstat + age , data = Boston )
lm.fit2
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Coefficients:
## (Intercept)        lstat          age
##    33.22276     -1.03207      0.03454
```

Summary

```r
summary(lm.fit2)
```

```
##
## 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.
##
```

Anova

```r
anova(lm.fit2)
```

```
## Analysis of Variance Table
##
## Response: medv
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## lstat       1 23243.9 23243.9 609.955 < 2.2e-16 ***
## age         1   304.3   304.3   7.984  0.004907 **
## Residuals 503 19168.1    38.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
```

The Boston data set contains 13 variables,

First check if all variables are numeric?

```r
sapply(Boston,class)
```

```
##      crim        zn     indus      chas       nox
## "numeric" "numeric" "numeric" "integer" "numeric" "numer
##       dis       rad       tax   ptratio     black      ls
## "numeric" "integer" "numeric" "numeric" "numeric" "numer
```

Note

It would be cumbersome to have to type all 13 variables in order to perform a regression using all of the predictors. Instead, we can use the following

```
lm.fit3 = lm(formula = medv ~ ., data = Boston )
lm.fit3
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Coefficients:
## (Intercept)         crim           zn        indus
##   3.646e+01   -1.080e-01    4.642e-02    2.056e-02    2.
##         nox           rm          age          dis
##  -1.777e+01    3.810e+00    6.922e-04   -1.476e+00    3.
##         tax      ptratio        black        lstat
##  -1.233e-02   -9.527e-01    9.312e-03   -5.248e-01
```

Summary

```r
summary (lm.fit3)
```

```
##
## 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 ***
```

We can access the individual components of a summary object by name

(type ?summary.lm to see what is available).

- Hence summary(lm.fit)$r.sq gives us the R2, and
- summary(lm.fit)$sigma gives us the RSE.

```
?summary.lm
```

```
## starting httpd help server ... done
```

```
summary (lm.fit3)$r.sq
```

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

```
summary(lm.fit3)$sigma
```

```
## [1] 4.745298
```

Variance Inflation Factors and Multicollinearity

The vif() function, part of the car package - Variance Inflation Factors can be used to compute variance inflation factors. Most VIF's are low to moderate for this data. Larger the VIF greater the multicollinearity.

The car package is not part of the base R installation so it must be downloaded the first time you use it via the install.packages option in R.

```
library(carData)
library(car)
vif(lm.fit3)
```

```
##     crim       zn    indus     chas      nox       rm
## 1.792192 2.298758 3.991596 1.073995 4.393720 1.933744 3.
##      rad      tax  ptratio    black    lstat
## 7.484496 9.008554 1.799084 1.348521 2.941491
```

Interpreting the Variance Inflation Factor

Variance inflation factors range from 1 upwards. The numerical value for VIF tells you (in decimal form) what percentage the variance is inflated for each coefficient. For example, a VIF of 1.9 tells you that the variance of a particular coefficient is 90% bigger than what you would expect if there was no multicollinearity

If there was no correlation with other predictors. A rule of thumb for interpreting the variance inflation factor:

- 1 = not correlated.
- Between 1 and 5 = moderately correlated.
- Greater than 5 = highly correlated.

$VIF = 1/1 - R_i^2$ where R_i^2 is the R^2 value when ith predictor is regressed against other predictors.

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

For example, in the above regression output, age has a high p-value. So we may wish to run a regression excluding this predictor. The following syntax results in a regression using all predictors except age.

```
lm.fit4=lm(medv~.-age ,data=Boston )
summary(lm.fit4)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
```

Alternatively, the update() function can be used.

```
lm.fit5=update(lm.fit3,~.-age)
summary(lm.fit5)
```

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

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 lstat×age as predictors; it is a shorthand for lstat+age+lstat:age.

NOTE: lm.fit6=lm(medv~lstat *age ,data=Boston))= lm (medv~ lstat+age+lstat:age ,data=Boston)

```
lm.fit6 = lm(medv~lstat *age ,data=Boston )
summary(lm.fit6)
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.806   -4.045   -1.333    2.085   27.552
```

Interaction Terms with no code

```
lm.fit6 = lm(medv~lstat *age ,data=Boston )
summary(lm.fit6)
```

```
##
## 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 *
## ---
```

Non-linear Transformations of the Predictors

The lm() function can also accommodate non-linear transformations of the predictors. For instance, given a predictor X, we can create a predictor X^2 using I(X^2). The function I() is needed since the ^ has a special meaning in a formula; We now perform a regression of medv onto lstat and $lstat^2$.

```
lm.fit7=lm(medv~lstat +I(lstat ^2))
summary (lm.fit7)
```

```
##
## 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:
```

Non-linear Transformations of the Predictors without code

```
##
## 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
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
```

Quadratic fit is superior to the linear fit

anova() function to further quantify the extent to which the quadratic fit is superior to the linear fit. The near-zero p-value associated with the quadratic term suggests that it leads to an improved model.

```
lm.fit =lm(medv~lstat)
anova(lm.fit,lm.fit7)
```

```
## 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
```

Outcome

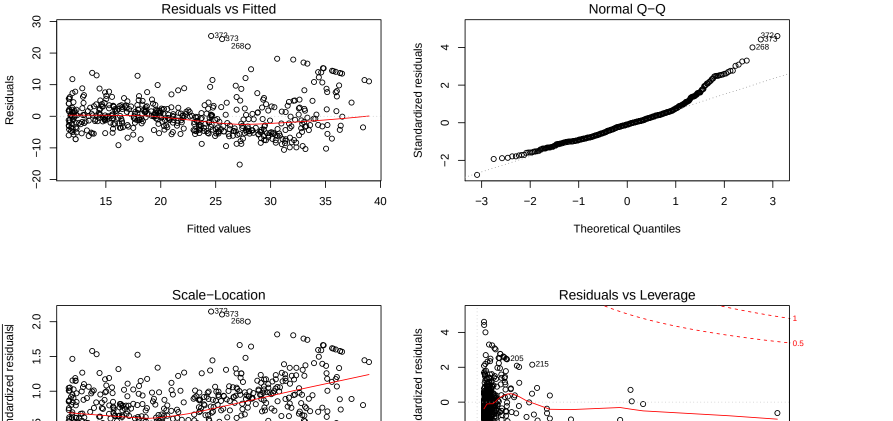
Here Model 1 represents the linear submodel containing only one predictor,lstat, while Model 2 corresponds to the larger quadratic model that has two predictors, lstat and lstat2. The anova() function performs a hypothesis test comparing the two models. The null hypothesis is that the two models fit 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.

Residual plots of the quadratic model

We see that when the lstat2 term is included in the model, there is little discernible pattern in the residuals.

```
par(mfrow=c(2,2))
plot(lm.fit7)
```



Higher Order Polynomials

In order to create a cubic fit, we can include a predictor of the form I(X^3). A better approach involves using the poly() function to create the polynomial within lm(). For example, the following command produces a fifth-order polynomial fit:

```
lm.fit9=lm(medv~poly(lstat ,5))
summary(lm.fit9)
```

```
##
## 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|)
```

Results

- This suggests that including additional polynomial terms, up to fifth order, leads to an improvement in the model fit!
- However, further investigation of the data reveals that no polynomial terms beyond fifth order have significant p-values in a regression fit.
- 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
##
## Residual standard error: 6.915 on 504 degrees of freedom
```

Qualitative Predictors

We will now examine the Carseats data, which is part of the ISLR library. We will attempt to predict Sales (child car seat sales) in 400 locations based on a number of predictors.

```
library(ISLR)
attach(Carseats )
names(Carseats )
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertis
##  [6] "Price"       "ShelveLoc"   "Age"         "Educatio
## [11] "US"
```

```
dim(Carseats)
```

```
## [1] 400  11
```

Check variable types

```r
sapply(Carseats,class)
```

```
##       Sales   CompPrice      Income Advertising  Populat
##   "numeric"   "numeric"   "numeric"   "numeric"   "numer
##   ShelveLoc         Age   Education       Urban
##    "factor"   "numeric"   "numeric"    "factor"    "fact
```

Qualitative Predictors

The Carseats data includes qualitative predictors such as Shelveloc, an indicator of the quality of the shelving location—that is, the space within a store in which the car seat is displayed—at each location.

The predictor Shelveloc takes on three possible values, Bad, Medium, and Good.

Given a qualitative variable such as Shelveloc, R generates dummy variables automatically. Below we fit a multiple regression model that includes some interaction terms.

```
lm.fit10 =lm(Sales~.+ Income:Advertising +Price:Age ,data=C
summary (lm.fit10)
```

```
##
## Call:
## lm(formula = Sales ~ . + Income:Advertising + Price:Age,
##
## Residuals:
```

Output without code

```
##
## Call:
## lm(formula = Sales ~ . + Income:Advertising + Price:Age,
##
## 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-1
## CompPrice          0.0929371  0.0041183  22.567  < 2e-1
## Income             0.0108940  0.0026044   4.183 3.57e-0
## Advertising        0.0702462  0.0226091   3.107 0.0020
## Population         0.0001592  0.0003679   0.433 0.6653
## Price             -0.1008064  0.0074399 -13.549  < 2e-1
## ShelveLocGood      4.8486762  0.1528378  31.724  < 2e-1
## ShelveLocMedium    1.9532620  0.1257682  15.531  < 2e-1
```

The contrasts() function

returns the coding that R uses for the dummy variables.

```
## The following objects are masked from Carseats (pos = 3
##
##     Advertising, Age, CompPrice, Education, Income, Popu
##     Price, Sales, ShelveLoc, Urban, US
```

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

NOTE: Use ?contrasts to learn about other contrasts, and how to set them.

- R has created a ShelveLocGood dummy variable that takes on a value of 1 if the shelving location is good, and 0 otherwise.
- It has also created a ShelveLocMedium dummy variable that equals 1 if the shelving location is medium, and 0 otherwise.
- A bad shelving location corresponds to a zero for each of the two dummy variables.
- The fact that the coefficient for ShelveLocGood in the regression output is positive indicates that a good shelving location is associated with high sales (relative to a bad location).
- And ShelveLocMedium has a smaller positive coefficient, indicating that a medium shelving location leads to higher sales than a bad shelving location but lower sales than a good shelving location.

Writing Functions

We now create the function. Note that the + symbols are printed by R and should not be typed in. The { symbol informs R that multiple commands are about to be input. Hitting Enter after typing { will cause R to print the + symbol. We can then input as many commands as we wish, hitting Enter after each one. Finally the } symbol informs R that no further commands will be entered.

- Craete a function called LoadLibraries:

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

Now if we type in LoadLibraries, R will tell us what is in the function.

```
## 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.

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