Statistical Learning Workshops: Linear Regression

Md Sakhawat Hossen

Former Data Analyst at Datasoft, Bangladesh
sakhawat3003@gmail.com

01/21/2022

Linear Regression

Linear regression is the stepping stone for building any statistical learning model or Machine Learning Model. Although, it is somewhat dull in comparison to other vastly popular and complex learning algorithm but Linear regression is still a very useful tool for predicting quantitative response. We can easily make good inference on which variables or predictors are highly responsible for driving the response in good faith. The term *linear* is not simply limited to applying only linear terms of variables as the predictors but also we can fit polynomial version of the variables as predictors to build more complex model to capture the behavior of data.

We will start with the *Simple Linear Regression* where we will make prediction based on a single variable. Later, we will introduce multiple linear regression to accommodate many predictors, and interaction terms in a single model.

Simple Linear Regression

For this workshop, we first need to install *ISLR2* library in the R environment if it is not installed beforehand. The *ISLR2* library includes several dataset including *Boston* and *Carseats* dataset which we will use in this workshop.

```
library(ISLR2)
data("Boston")
```

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

Boston dataset is the slightly modified version of the *Boston* dataset from the “MASS” library in *R*.

The dataset contains housing values in 506 suburbs of Boston. It is a data frame with 506 rows and 13 variables. To know more about the dataset we can type by *?Boston* command.

We will fit the linear regression model with *medv*(median house value) as the response and *lstat*(lower status in the neighborhood) as the only predictor.

```
attach(Boston) #the variables in the Boston dataset will be available without calling it
#explicitly if we attach the dataset.
```

```
lm.fit<-lm(medv~lstat) #fitting the simple linear model with just one variable as predictor
lm.fit
```

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

The call on *lm.fit* only reveals minimal information: Intercept and coefficient for *lstat*.

We can use the *summary* function for more detailed information including p-values and standard errors for the coefficients, the R^2 and F statistics for the fitted 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
```

```
names(lm.fit) #names of the available information in the model.
```

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

We can use the above names in the model following a dollar sign to extract any particular information, or we can use other built-in functions in R to extract the information from the fitted model. Examples are given below:

```
lm.fit$coefficients #gives the intercept and the coefficients
```

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

```
coef(lm.fit) #using coef function give the same result
```

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

```
confint(lm.fit) #give the 95% confidence interval for the intercept and coefficients
```

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

The *predict* function is the primary option to predict any outcome from the model given the *lstat* value. The prediction can be accompanied by the confidence interval and the prediction interval. Confidence interval is mostly expected when we want to predict the average outcome from the model and prediction interval for predicting a single outcome from a single input.

```
predict(object = lm.fit, newdata = 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
```

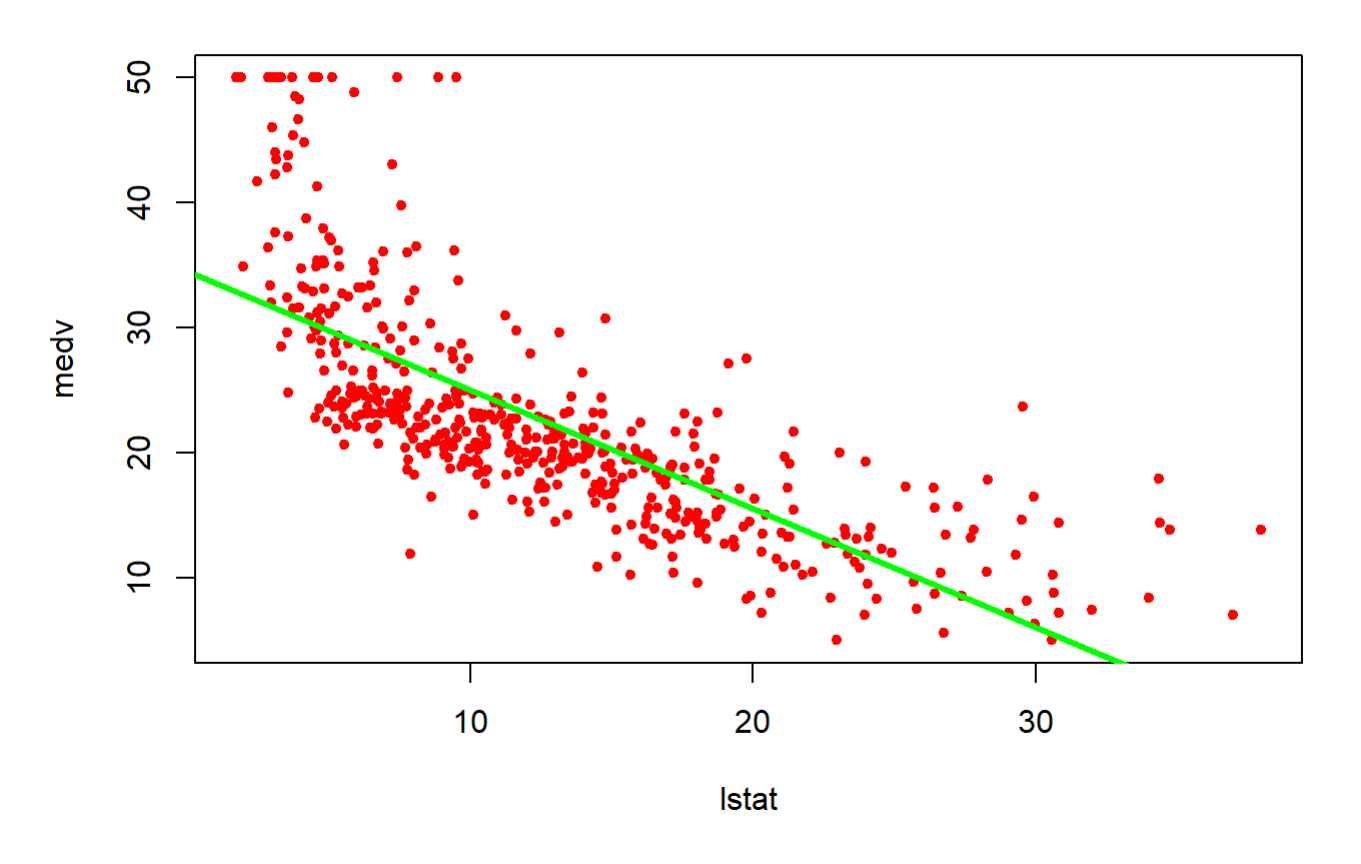
Here, we see the output for each of the lstat value provided in the data frame. For each of the predicted outcome, a 95% confidence interval is provided around the predicted response.

```
predict(object = lm.fit, newdata =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
```

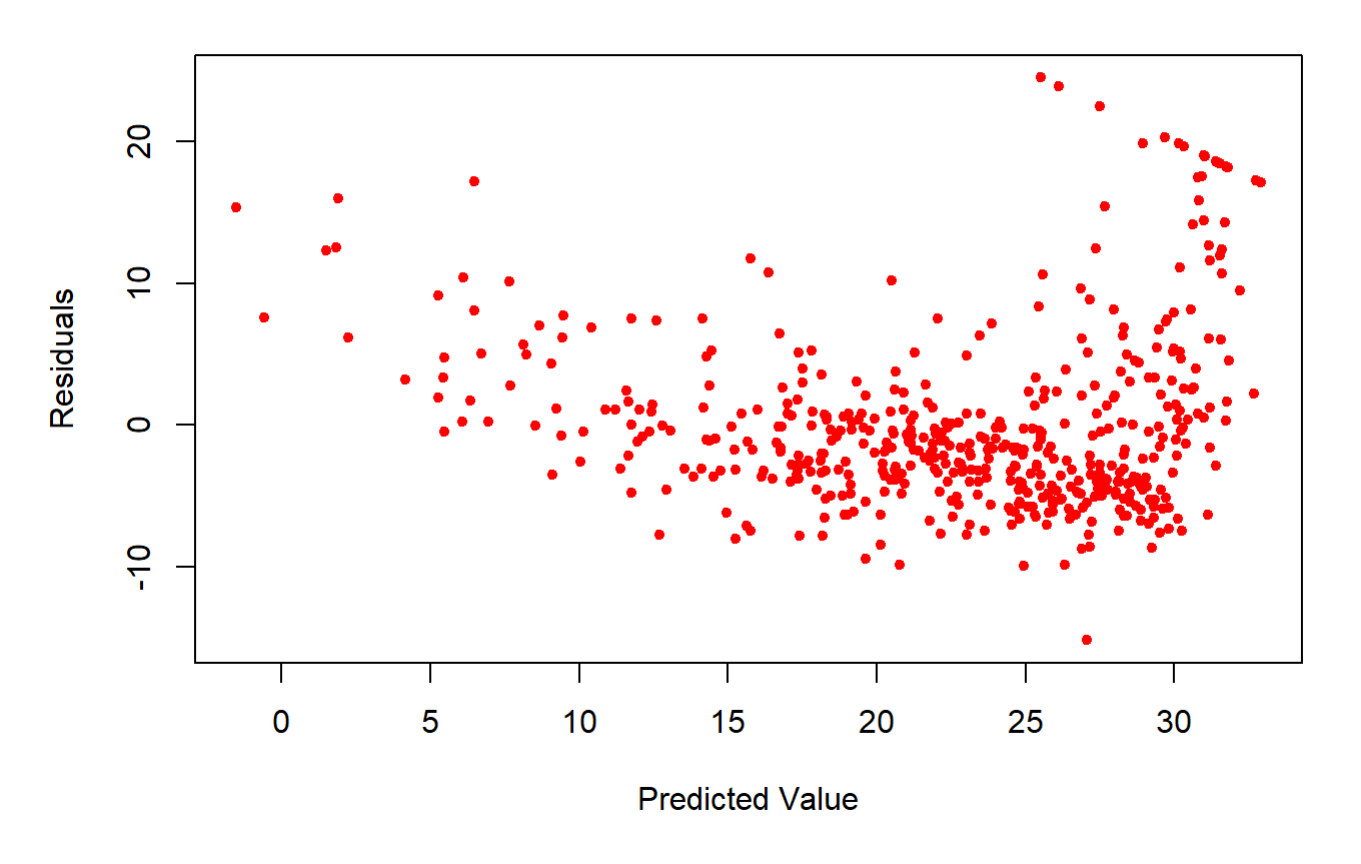
The predicted responses are now provided with the prediction intervals. Prediction interval is much more wider than the confidence interval as it accounts for the irreducible errors in the prediction.
Now, we will plot our least square regression line along with the actual data to manifest how our model was fitted.

```
plot(lstat,medv, pch=20, col="red") #plotting the actual data (red dots)
abline(lm.fit, lwd=3, col="green") #drawing the least square regression line(green) through the data
```

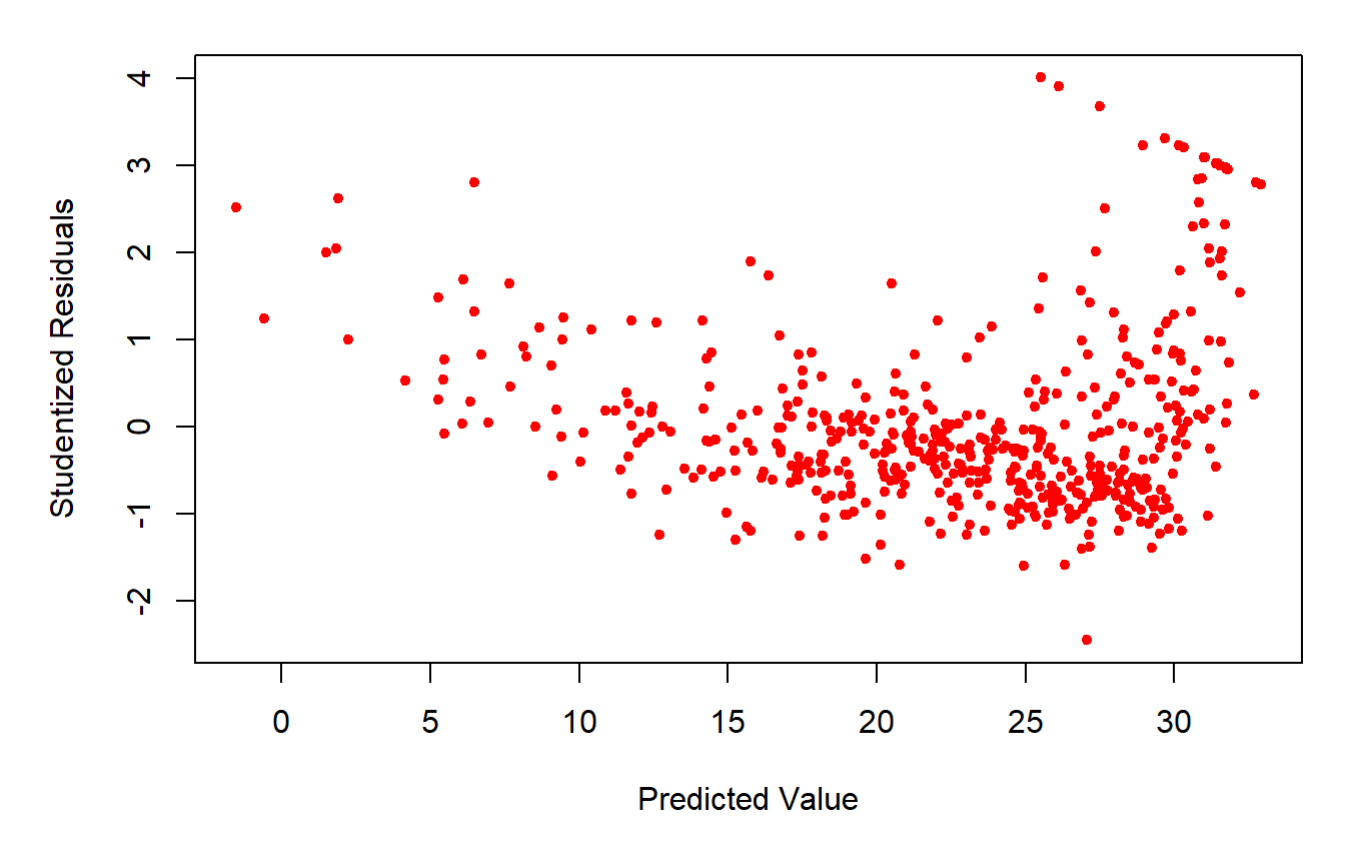


From the plot, we see the presence of some non-linearity in the relationship between *lstat* and *medv*.
We can also plot the residuals from a linear regression fit using the *residuals* function and *rstudent* function which returns the studentized residuals.

```
plot(predict(lm.fit), residuals(lm.fit), pch=20, col="red", xlab = "Predicted Value",
     ylab = "Residuals")
```




```
plot(predict(lm.fit), rstudent(lm.fit), pch=20, col="red", xlab = "Predicted Value",
     ylab = "Studentized Residuals")
```



Both the residuals and studentized residuals plots suggest strong evidence of non-linearity.

Multiple Linear Regression

We will again use the *lm* function to fit a linear regression model with multiple variables. For simplicity, we will fit the linear model with two variables, *lstat age*.

```
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
```

There are 12 variables in the *Boston* dataset. It will not be easy to type all the variable names to fit the model with all the variables. So, there is a shortcut to accommodate all the variables.

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

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1304  -2.7673  -0.5814   1.9414  26.2526
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.617270   4.936039   8.431 3.79e-16 ***
## crim         -0.121389   0.033000  -3.678 0.000261 ***
## zn            0.046963   0.013879   3.384 0.000772 ***
## indus         0.013468   0.062145   0.217 0.828520
## chas          2.839993   0.870007   3.264 0.001173 **
## nox         -18.758022   3.851355  -4.870 1.50e-06 ***
## rm            3.658119   0.420246   8.705  < 2e-16 ***
## age           0.003611   0.013329   0.271 0.786595
## dis          -1.490754   0.201623  -7.394 6.17e-13 ***
## rad           0.289405   0.066908   4.325 1.84e-05 ***
## tax          -0.012682   0.003801  -3.337 0.000912 ***
## ptratio      -0.937533   0.132206  -7.091 4.63e-12 ***
## lstat        -0.552019   0.050659 -10.897  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.798 on 493 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7278
## F-statistic: 113.5 on 12 and 493 DF,  p-value: < 2.2e-16
```

This is possible to access various components of *summary* object by name. We can check them out by the command *?summary.lm*

```
summary.fit<-summary(lm.fit)
summary.fit$r.squared #gives the R-Square
```

```
## [1] 0.734307
```

```
summary.fit$sigma #gives the RSE
```

```
## [1] 4.798034
```

Variance inflation factor(VIF) is an important measurement criteria for multi-colinearity in the predictor variables. As a rule of thumbs, VIF values greater than 5 should be considered as the presence of multi-colinearity in the variables. We can use the *vif* function from the *car* library for this purpose.

```
library(car)
```

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

```
vif(lm.fit)
```

```
##     crim       zn    indus     chas      nox       rm      age      dis
## 1.767486 2.298459 3.987181 1.071168 4.369093 1.912532 3.088232 3.954037
##      rad      tax  ptratio    lstat
## 7.445301 9.002158 1.797060 2.870777
```

As we can see, most of the VIF values are less than 5.

If we want to fit the model with all the variables except one or two then we can simply drop the variables. From our previous model, we have seen the p-value for the predictor *age* is high. So, we can fit the model again by eliminating the *age* variable.

```
lm.fit01<- lm(medv~.-age, data = Boston)
summary(lm.fit01)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1851  -2.7330  -0.6116   1.8555  26.3838
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.525128   4.919684   8.441 3.52e-16 ***
## crim         -0.121426   0.032969  -3.683 0.000256 ***
## zn            0.046512   0.013766   3.379 0.000785 ***
## indus         0.013451   0.062086   0.217 0.828577
## chas          2.852773   0.867912   3.287 0.001085 **
## nox         -18.485070   3.713714  -4.978 8.91e-07 ***
## rm            3.681070   0.411230   8.951  < 2e-16 ***
## dis          -1.506777   0.192570  -7.825 3.12e-14 ***
## rad           0.287940   0.066627   4.322 1.87e-05 ***
## tax          -0.012653   0.003796  -3.333 0.000923 ***
## ptratio      -0.934649   0.131653  -7.099 4.39e-12 ***
## lstat        -0.547409   0.047669 -11.483  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.794 on 494 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7284
## F-statistic: 124.1 on 11 and 494 DF,  p-value: < 2.2e-16
```

Instead of fitting the model from scratch, we can simply update the model.

```
lm.fit01<-update(lm.fit, ~.-age) #updating without the age variable
```

Interaction Term

The inclusion of interaction terms in linear regression setting begs attention for good reasons. Interaction terms serve the purpose of *Synergy* effect: increasing or decreasing one predictor variable also influences the dynamics of another predictor variable.

The syntax *lstat:age* tells R to include an interaction term between the predictor variables *lstat* and *age*. We can also do the same thing with the syntax $lstat * age$ which accommodates not only the predictor variables in the model but also the interaction term. It is a shorthand formula for $lstat + age + lstat : age$

```
fit.interaction<-lm(medv~lstat*age, data = Boston)
summary(fit.interaction)
```

```
##
## 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 Transformation of the Predictors

In the previous model, we have seen the data shows the prevalence of non-linearity when we plotted residuals against the predicted values. It showed discernible pattern which is not expected when we have good a linear relationship between the response and the predictor.

R facilitates the transformation of predictor to accommodate non-linear relationship through *lm* function. Here, we will build the model with *medv* as the response and $lstat + lstat^2$ as the predictor.

```
lm.fit2<-lm(medv~lstat+I(lstat^2), data = Boston) #here I() is a wrapper around the quadratic term
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
```

The R^2 has definitely improved and the near zero p-value for the coefficient of $lstat^2$ suggests significant improvement over the previous model where we included only linear terms.

Model Comparison: Anova test

We will further investigate, whether this model is superior than the previous linear model with no quadratic term. We will perform an *Anova* test to quantify the extent to which the quadratic fit is superior to the linear fit.

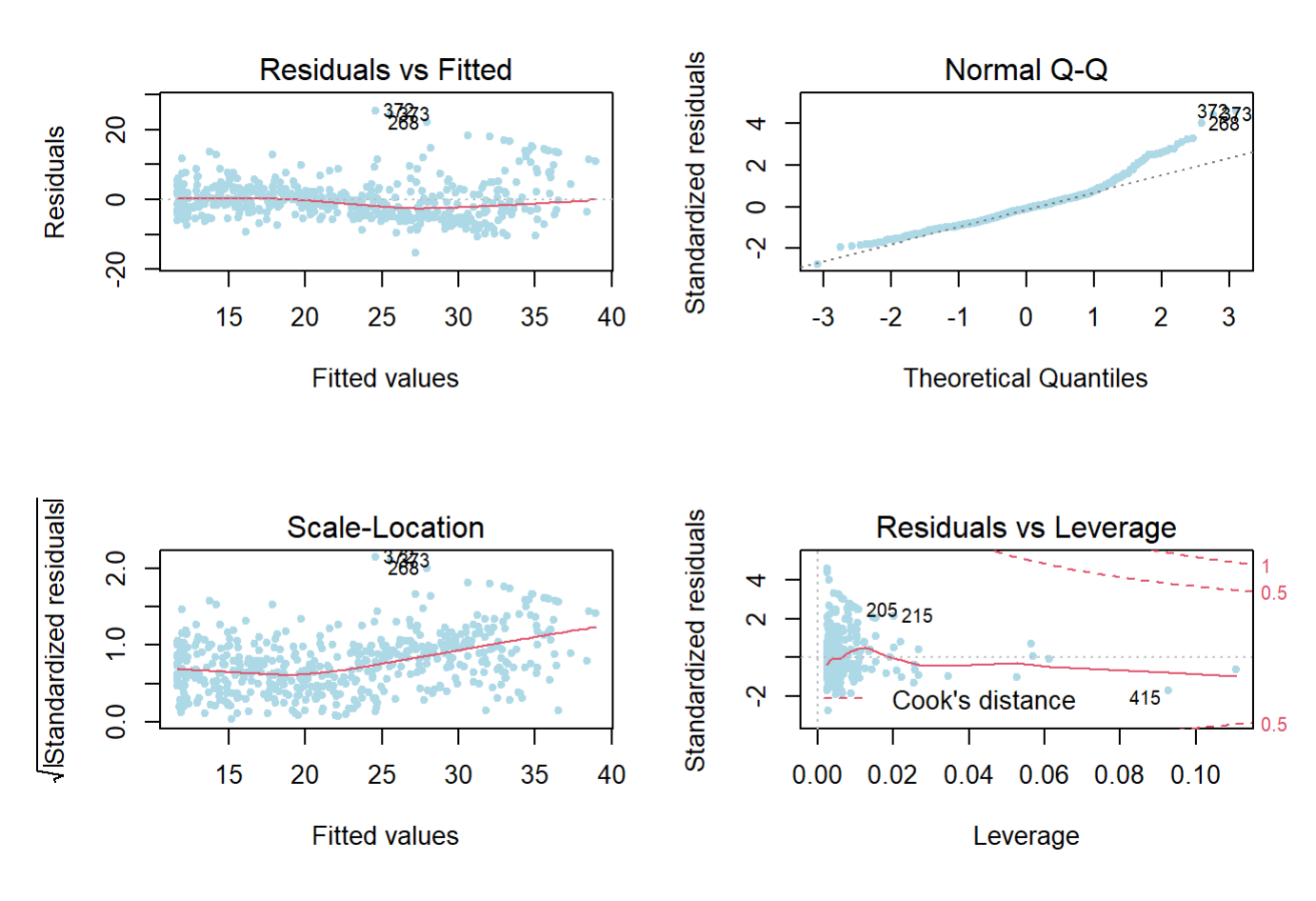
```
lm.fit<-lm(medv~lstat, data = Boston)
anova(lm.fit,lm.fit2)
```

```
## 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
```

The *anova* function here performs a hypothesis test between these two models. The null hypothesis is that the two models fit the data equally well. But, the alternative hypothesis is that the full model is superior than the simpler one.

We see a F-statistic of 135 and a p-value very close to zero for this hypothesis test. We can confirm the superiority of the model with $lstat^2$

```
par(mfrow=c(2,2))
plot(lm.fit2, col="light blue", pch=20)
```



From the plotted data of the model with $lstat^2$, we can see less discernible pattern of the residuals.

We can also accommodate higher order polynomials in the model by using the *poly* function.

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

```
##
## 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
```

Certainly, the model improves with the inclusion of polynomials up to 5 degree. All of those transformed polynomial predictors have very small p-values but further investigation suggests that the polynomials beyond 5 degree does not actually improve the model.

By default, the *poly()* function orthogonalizes the predictors: this means that the features output by this function are not simply a sequence of powers of the argument. However, a linear model applied to the output of the *poly()* function will have the same fitted values as a linear model applied to the raw polynomials (although the coefficient estimates, standard errors, and p-values will differ). In order to obtain the raw polynomials from the *poly()* function, the argument *raw = TRUE* must be used.

Qualitative Predictors

The *ISLR2* library contains another dataset: *Carseats*. We will try to estimate or predict the number of child seat sales in 400 different locations based on several predictors.

```
data("Carseats")
head(Carseats)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

This dataset contains categorical variables or qualitative predictors like *Shelveloc*, an indicator for the quality of the position of car seats inside the store. The predictor *Shelveloc* takes on three different categories: Good, Medium, and Bad. While fitting the model, R will automatically create dummy variables for these categories in the qualitative predictor.

We will fit a regression model with all the variables including a couple of interaction terms as well to incorporate the synergy effect.

```
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
```

We can see, R has created a dummy variable *ShelvelocGood* that takes on a value of 1 if the location is good inside the store and another dummy variable *ShelvelocMedium* that is 1 for the medium location and 0 otherwise. The bad shelving location is serving as the reference and equal to zero for each of the two dummy variables created. The coefficient of the *ShelvelocGood* in the fitted linear model is good indicating a higher sale for the good location. The coefficient for *ShelvelocMedium* is low but still positive indicating a lower sale than the good location but still higher sale than a bad shelving location.