Linear Regression in R

DATA621 Blog 02

Zhi Ying Chen

27 November 2020

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they-indicated by the magnitude and sign of the beta estimates-impact the outcome variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable. The model fits a line that is closest to all observation in the dataset. The basic assumption here is that functional form is the line and it is possible to fit the line that will be closest to all observation in the dataset.

Load Packages

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.5.3
```

```
library(ggplot2)
library(caTools)
```

Read Data

```
set.seed(3)
#Check variable types
sapply(Boston, class)
```

```
##      crim        zn     indus      chas       nox        rm       age       dis
## "numeric" "numeric" "numeric" "integer" "numeric" "numeric" "numeric" "numeric"
##       rad       tax   ptratio     black     lstat      medv
## "integer" "numeric" "numeric" "numeric" "numeric" "numeric"
```

```
#Summarize variables
summary(Boston)
```

```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
```

```
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

I split Boston dataset as 80% as training set, and 20% as testing set and make the model for the training dataset. It can be seen that training dataset has 404 observations and testing dataset has 102 observations.

```
set.seed(100)
sample <- sample(1:nrow(Boston), 0.8*nrow(Boston))
train = Boston[sample,]
test = Boston[-sample,]
dim(train)
```

```
## [1] 404  14
```

```
dim(test)
```

```
## [1] 102  14
```

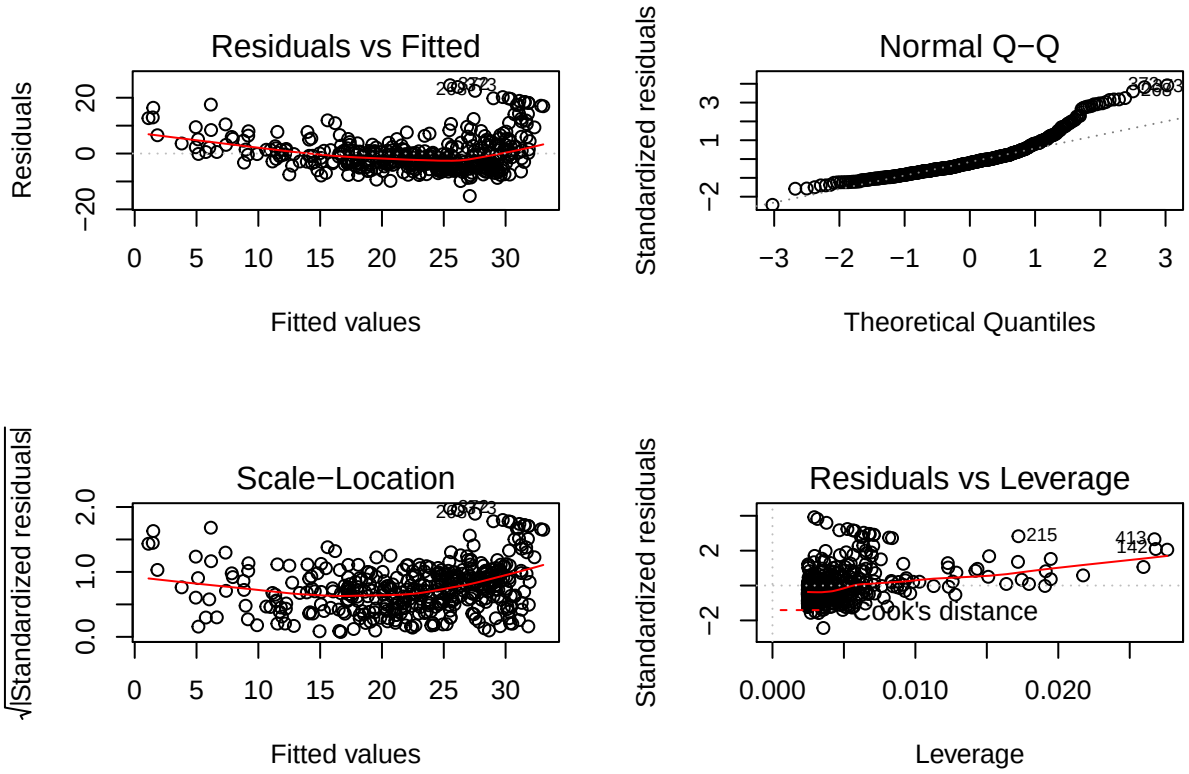
Model: Simple Linear Regression Model Results

```
model_1 = lm(medv~lstat, data=train)
summary(model_1)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.204  -4.002  -1.363   2.055  24.491
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.72056    0.63868   54.36   <2e-16 ***
## lstat       -0.96657    0.04459  -21.68   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.259 on 402 degrees of freedom
## Multiple R-squared:  0.539,  Adjusted R-squared:  0.5378
## F-statistic:   470 on 1 and 402 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(model_1)
```



Observation from summary

1, Is there a relationship between predictor and response variables?

We can answer this using F stats which defines the collective effect of all predictor variables on the response variable. In this model, F = 470 is far greater than 1, and so it can be concluded that there is a relationship between predictor and response variable.

2, Is this model fit?

We can answer this based on R2 (multiple-R-squared) value as it indicates how much variation is captured by the model. R2 closer to 1 indicates that the model explains the large value of the variance of the model and hence a good fit. In this case, the value is 0.539 (not really closer to 1) and hence the model may not a good fit.

Confidence Intervals and Predictions

We want to know something about the confidence intervals of our coefficients and/or we might want to use our model to make some predictions. The confint() function Computes confidence intervals for one or more parameters in a fitted model. And the predict() function can be utilized to produce both confidence and prediction intervals for the prediction of medv for a given value of lstat.

```
confint(model_1)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.464985 35.9761331
## lstat       -1.054217 -0.8789134
```

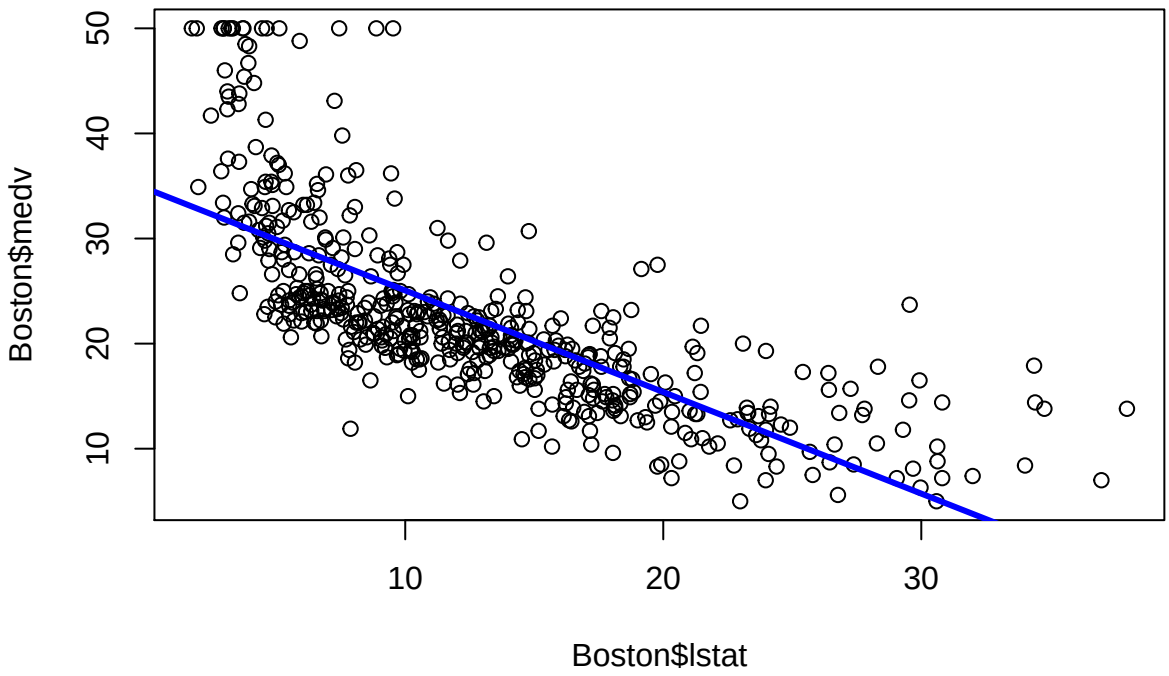
```
predict(model_1, data.frame(lstat=c(5,10,15,20)),interval="confidence")
```

```
##        fit      lwr      upr
## 1 29.88773 28.98902 30.78644
## 2 25.05491 24.40449 25.70532
## 3 20.22208 19.57205 20.87211
## 4 15.38926 14.49138 16.28713
```

```
predict(model_1, data.frame(lstat=c(5,10,15,20)),interval="prediction")
```

```
##        fit       lwr      upr
## 1 29.88773 17.550239 42.22523
## 2 25.05491 12.733011 37.37680
## 3 20.22208  7.900206 32.54396
## 4 15.38926  3.051822 27.72669
```

```
plot(Boston$lstat, Boston$medv)
abline(model_1, lwd=3, col="blue")
```



Conclusion

The example shows how to approach linear regression modeling. The model that is created still has scope for improvement as we can apply techniques like Outlier detection, Correlation detection to further improve the accuracy of more accurate prediction.