Lab: Linear Regression

Contents

3.6.1 Libraries 3

3.6.2 Simple Linear Regression 3

Description of Variables . . . 3
Fitting a Simple Linear Regression Model . . . 3
check information contained in lm.fit object . . . 4
extracting the quantities . . . 4
Finding the confidence interval . . . 4
Predict Function . . . 4
Plotting the results . . . 5
Diagnostics Plot . . . 6
Residuals vs Fitted Values . . . 6
Standardized Residuals vs Fitted Values . . . 7
Calculating hatvalues . . . 8

3.6.3 Multiple Linear Regression 9

Fiting A Multiple Linear Regression with lstat and age as predictors . . . 9
Fitting the Full model with all of the predictors . . . 10
Extracting the components of the summary of the linear model . . . 11
Extrating R-squared . . . 11
Extracting RSE . . . 11
all the available information in the summary object . . . 11
Variance Inflation Factor(VIF) . . . 11

3.6.4 Interaction Terms 12

3.6.5 Non-linear Transformation of the Predictors 13

ANOVA . . . 14
Diagnostic Plots . . . 14
Adding polynomial terms to the data . . . 15
log transformation . . . 16

3.6.6 Qualitative Predictors **16**
Regression Model 17

3.6.1 Libraries

We load the MASS and ISLR package

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

3.6.2 Simple Linear Regression

We will be using the Boston Dataset from the MASS package.

```
data <- Boston
kableExtra::kable(head(data), digits = 3)
```

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.006	18	2.31	0	0.538	6.575	65.2	4.090	1	296	15.3	396.90	4.98	24.0
0.027	0	7.07	0	0.469	6.421	78.9	4.967	2	242	17.8	396.90	9.14	21.6
0.027	0	7.07	0	0.469	7.185	61.1	4.967	2	242	17.8	392.83	4.03	34.7
0.032	0	2.18	0	0.458	6.998	45.8	6.062	3	222	18.7	394.63	2.94	33.4
0.069	0	2.18	0	0.458	7.147	54.2	6.062	3	222	18.7	396.90	5.33	36.2
0.030	0	2.18	0	0.458	6.430	58.7	6.062	3	222	18.7	394.12	5.21	28.7

Description of Variables

crim: per capita crime rate by town.
zn: proportion of residential land zoned for lots over 25,000 sq.ft.
indus: proportion of non-retail business acres per town.
chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
nox: nitrogen oxides concentration (parts per 10 million).
rm: average number of rooms per dwelling.
age: proportion of owner-occupied units built prior to 1940.
dis: weighted mean of distances to five Boston employment centres.
rad: index of accessibility to radial highways.
tax: full-value property-tax rate per $10,000.
ptratio: pupil-teacher ratio by town.
black: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
lstat: lower status of the population (percent).
medv: median value of owner-occupied homes in $1000s.

Fitting a Simple Linear Regression Model

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

```
##
## Call:
```

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

check information contained in lm.fit object

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

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

extracting the quantities

Here we extract the coefficients of the regression model.

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

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

Finding the confidence interval

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

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

Predict Function

The predict function is used to predict the dependent values using the regression model. It also gives the confidence or the prediction interval.

Prediction with Confidence Interval

```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
```

Prediction with Prediction Interval

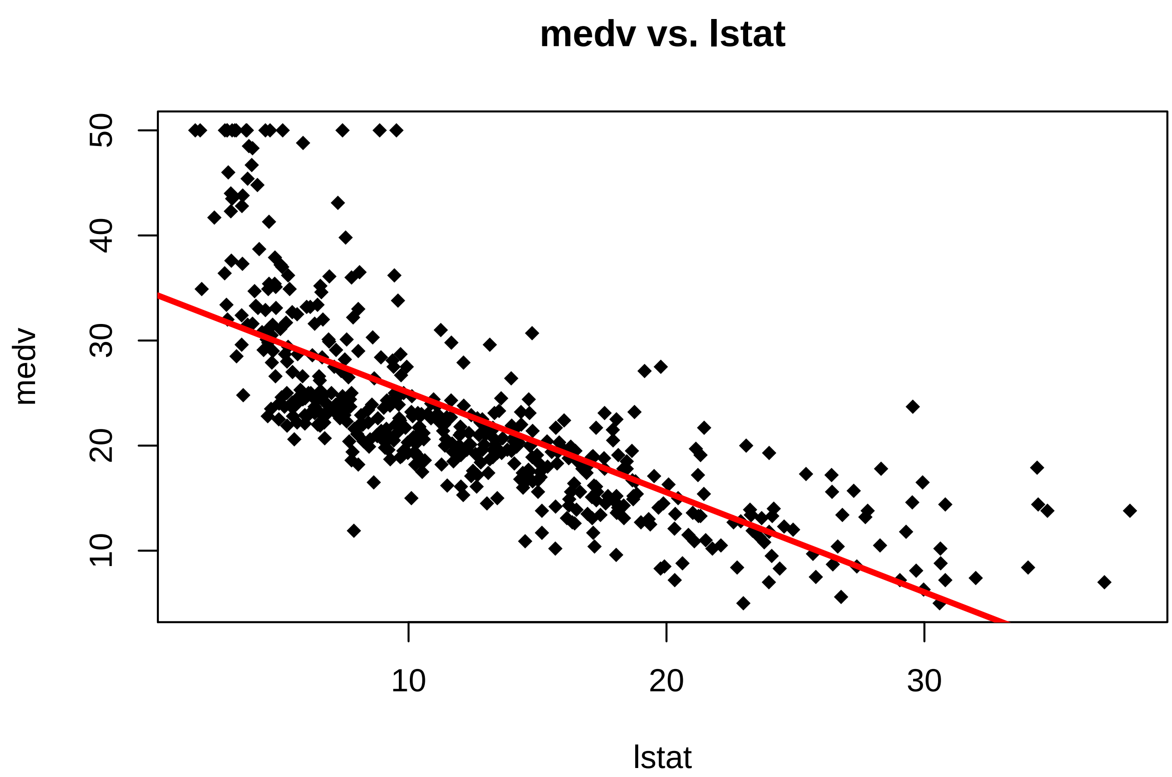
```r
predict(lm.fit, data.frame(lstat = c(5,10,15)), interval = 'prediction')
```

```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

As we can see, the prediction interval is larger than the confidence interval. This is due to prediction interval containing the uncertainity associated with the irreducible error.

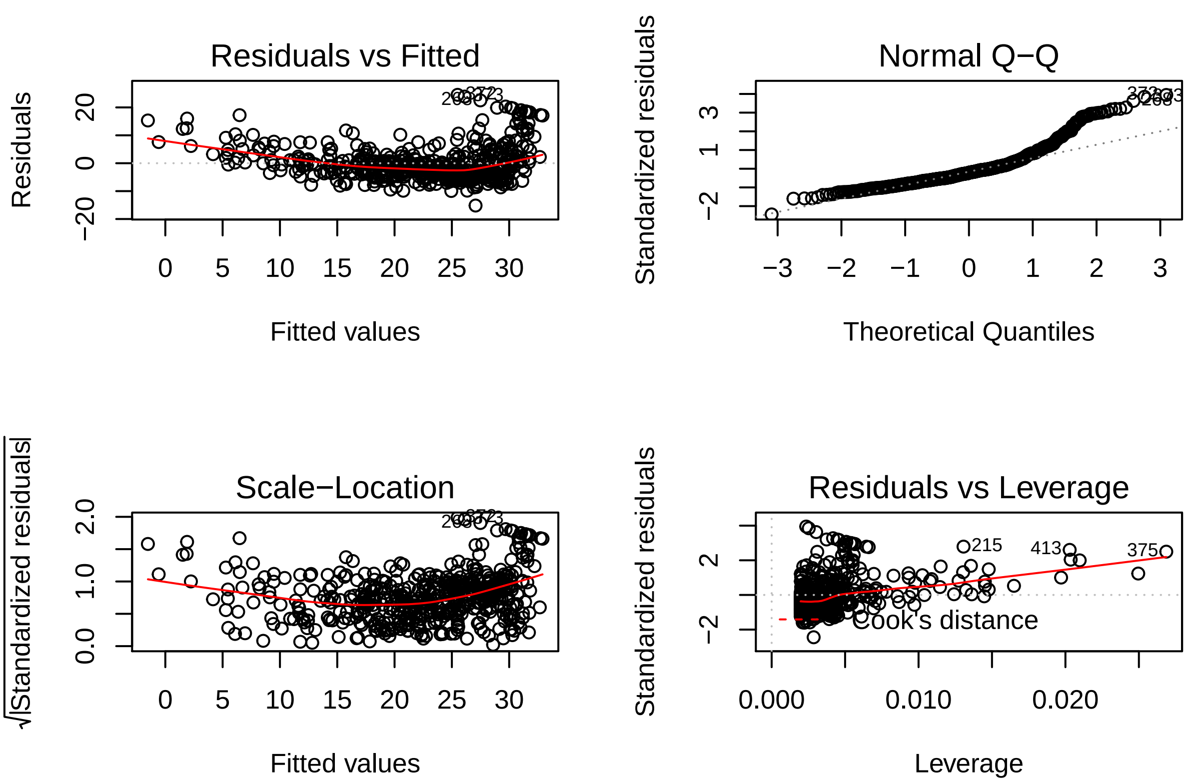
Plotting the results

```r
plot(data$lstat,data$medv, xlab = 'lstat', ylab = 'medv', main = 'medv vs. lstat', pch =18)
abline(lm.fit,lwd = 3, col = 'red')
```



Diagnostics Plot

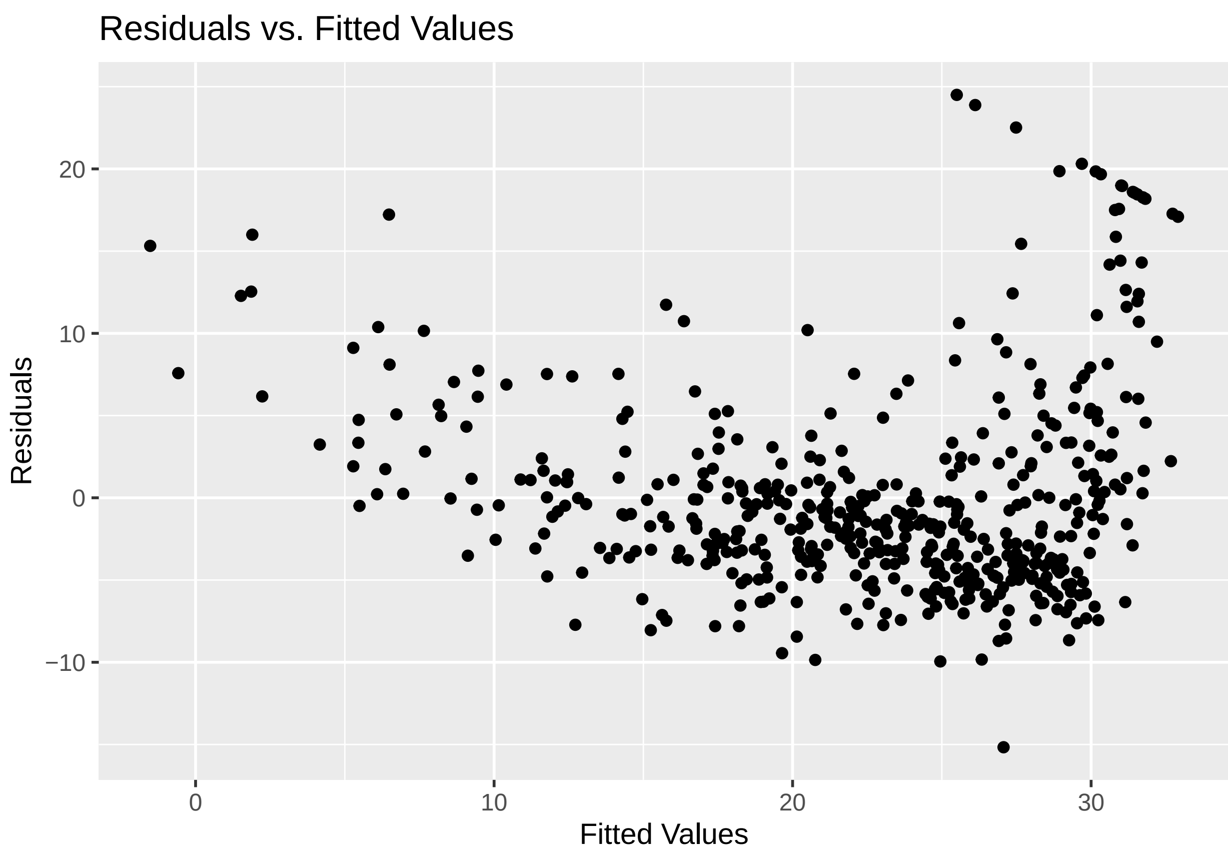
```r
par(mfrow = c(2,2))
plot(lm.fit)
```



Remarks
1. The Residuals vs Fitted Values plot shows some pattern. Thus, the residuals are not randomly distributed.
2. The Q-Q plot also has some problems towards the ends.
3. The Residuals vs. Leverage plot shows the presence of bad leverage points. We will have to check the data points 215, 413 and 375.

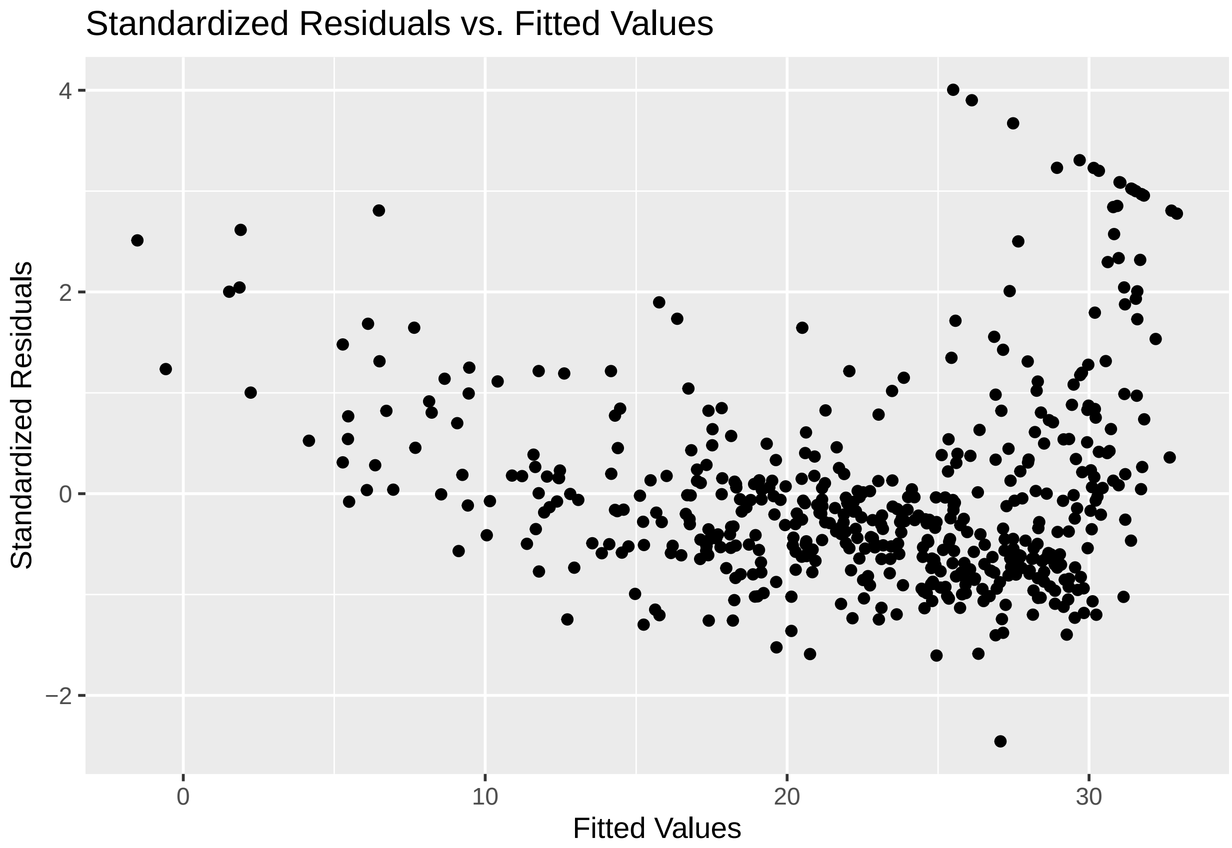
Residuals vs Fitted Values

```r
library(ggplot2)
ggplot(data = data.frame(fit = predict(lm.fit), residuals = residuals(lm.fit)),
                aes(x = fit,y = residuals)) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  ggtitle("Residuals vs. Fitted Values")
```



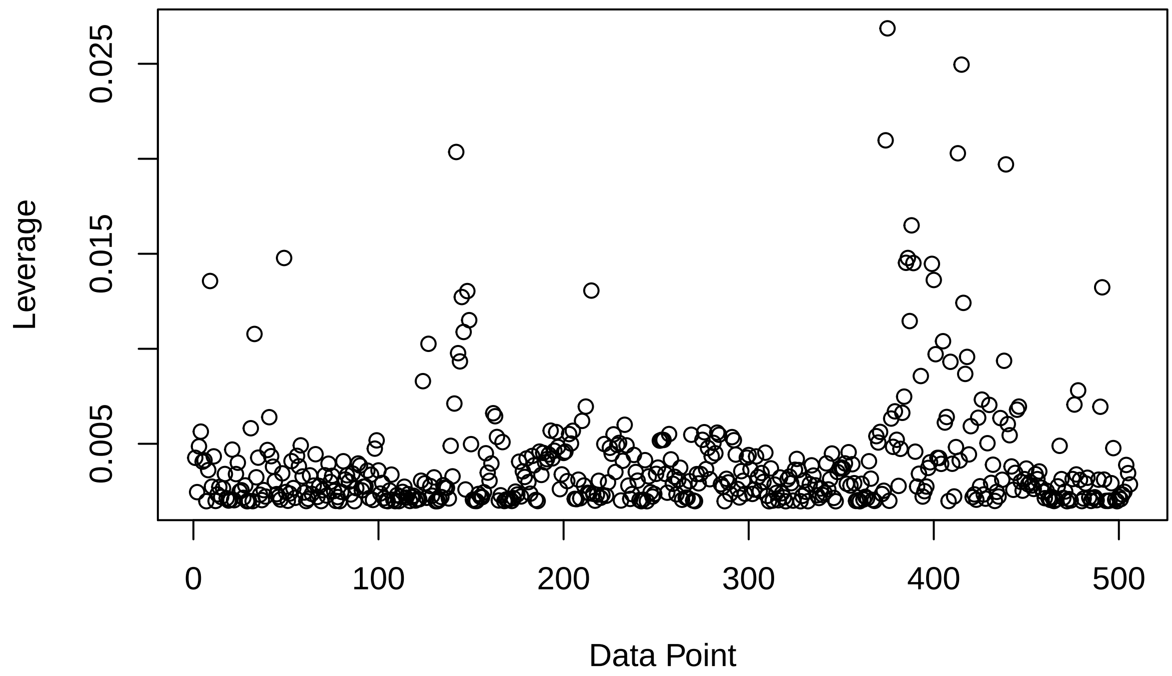
Standardized Residuals vs Fitted Values

```
ggplot(data = data.frame(fit = predict(lm.fit), residuals = rstudent(lm.fit)),
                aes(x = fit,y = residuals)) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Standardized Residuals") +
  ggtitle("Standardized Residuals vs. Fitted Values")
```



Calculating hatvalues

```r
plot(hatvalues(lm.fit), xlab = 'Data Point', ylab = 'Leverage')
```




```
which.max(hatvalues(lm.fit))
```

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

3.6.3 Multiple Linear Regression

Fiting A Multiple Linear Regression with lstat and age as predictors

```
lm.fit = lm(medv ~ lstat + age, data = data)
summary(lm.fit)
```

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

Fitting the Full model with all of the predictors

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

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

From the above summaey we see that indus and age has high p-value. Hence, they are not statistically significant.

Extracting the components of the summary of the linear model

```r
#saving the summary into another object
sum <- summary(lm.fit)
```

Extrating R-squared

```r
sum$r.squared
```

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

Extracting RSE

```r
sum$sigma
```

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

all the available information in the summary object

```r
names(sum)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

Variance Inflation Factor(VIF)

The library Car has a function to get the VIF.

```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
```

tax has a VIF greater than 5. This means that multicollinearity is affecting the model.

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

```
##
## Call:
## lm(formula = medv ~ . - age - indus, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.5984  -2.7386  -0.5046   1.7273  26.2373
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.341145   5.067492   7.171 2.73e-12 ***
## crim         -0.108413   0.032779  -3.307 0.001010 **
## zn            0.045845   0.013523   3.390 0.000754 ***
## chas          2.718716   0.854240   3.183 0.001551 **
## nox         -17.376023   3.535243  -4.915 1.21e-06 ***
## rm            3.801579   0.406316   9.356  < 2e-16 ***
## dis          -1.492711   0.185731  -8.037 6.84e-15 ***
## rad           0.299608   0.063402   4.726 3.00e-06 ***
## tax          -0.011778   0.003372  -3.493 0.000521 ***
## ptratio      -0.946525   0.129066  -7.334 9.24e-13 ***
## black         0.009291   0.002674   3.475 0.000557 ***
## lstat        -0.522553   0.047424 -11.019  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.736 on 494 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7348
## F-statistic: 128.2 on 11 and 494 DF,  p-value: < 2.2e-16
```

```r
vif(lm.fit1)
```

```
##     crim       zn     chas      nox       rm      dis      rad      tax
## 1.789704 2.239229 1.059819 3.778011 1.834806 3.443420 6.861126 7.272386
##  ptratio    black    lstat
## 1.757681 1.341559 2.581984
```

Still tax has a high VIF. We will need to perform variable selection on this model.

3.6.4 Interaction Terms

In this section, we will see how to add interaction terms.

```r
lm.fit2 <- lm(medv ~ lstat*age, data = data)
# lstat*age adds three variables to the model. lstat + age + lstat*age
# lstat:age addas only lstat*age
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = data)
##
## 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 Transformation of the Predictors

The lm function can also be used to add non-linear transformations of the predictor variables.

```r
lm.fit3 <- lm(medv ~ lstat + I(lstat^2), data = data)
summary(lm.fit3)
```

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

From the summary, we see that this model is valid. Now, we will perform ANOVA.

ANOVA

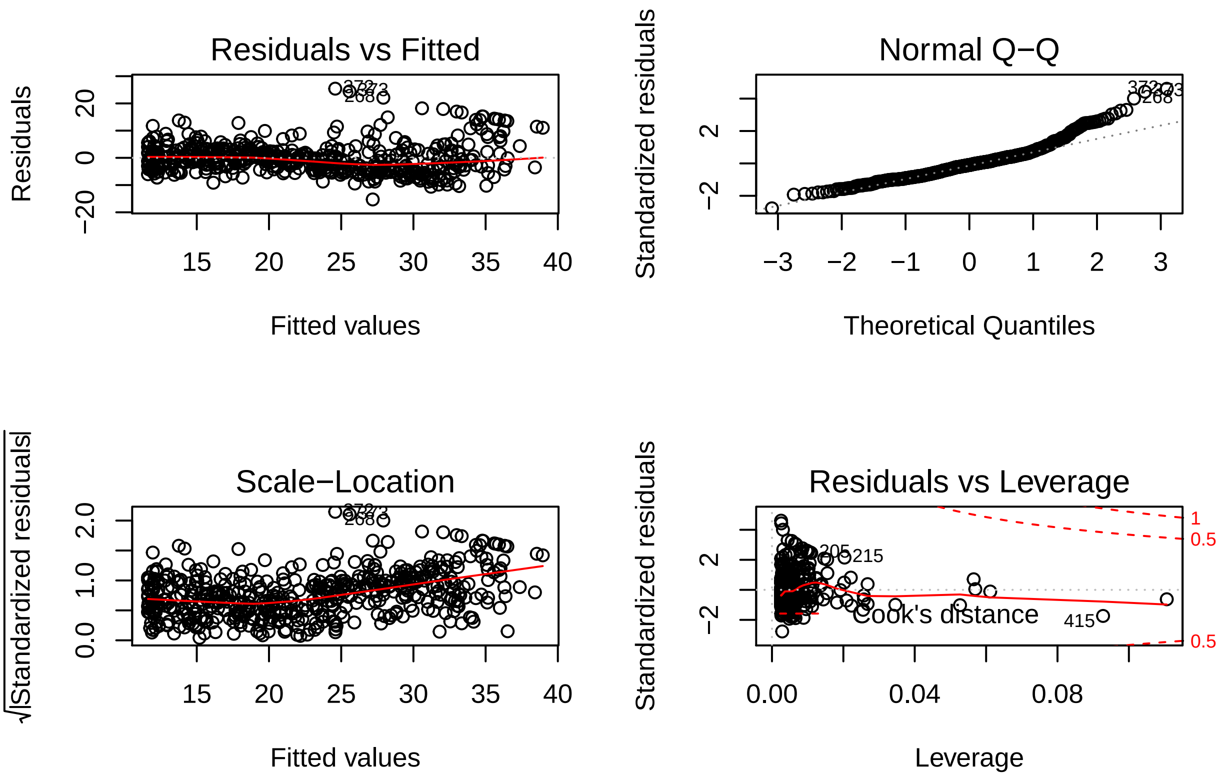
```r
#reduced model
lm.fit4 <- lm(medv ~ lstat, data = data)
anova(lm.fit4,lm.fit3)
```

```
## 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
```

Thus, we reject the null hypothesis that both model perform equally well.

Diagnostic Plots

```r
par(mfrow = c(2,2))
plot(lm.fit3)
```



Remarks
1. The Residuals vs Fitted values is better than our previous models. Now there is no apparent pattern in the residuals.
2. The Q-Q plot is similar to before.

Adding polynomial terms to the data

Use the poly(var, degree) syntax in lm to add polynomial terms in the model.

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

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

```
lm.fit6 <- lm(medv ~ poly(lstat,6),data = data)
summary(lm.fit6)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 6), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.7317  -3.1571  -0.6941   2.0756  26.8994
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.5328     0.2317  97.252  < 2e-16 ***
## poly(lstat, 6)1 -152.4595     5.2119 -29.252  < 2e-16 ***
## poly(lstat, 6)2   64.2272     5.2119  12.323  < 2e-16 ***
```

```
## poly(lstat, 6)3  -27.0511     5.2119  -5.190 3.06e-07 ***
## poly(lstat, 6)4   25.4517     5.2119   4.883 1.41e-06 ***
## poly(lstat, 6)5  -19.2524     5.2119  -3.694 0.000245 ***
## poly(lstat, 6)6    6.5088     5.2119   1.249 0.212313
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.212 on 499 degrees of freedom
## Multiple R-squared:  0.6827, Adjusted R-squared:  0.6789
## F-statistic: 178.9 on 6 and 499 DF,  p-value: < 2.2e-16
```

So it seems adding more terms than 5 order term is not useful.

log transformation

In this section we will use log transformation.

```r
lm.fit7 <- lm(medv ~ log(lstat), data = data)
summary(lm.fit7)
```

```
##
## Call:
## lm(formula = medv ~ log(lstat), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.4599  -3.5006  -0.6686   2.1688  26.0129
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  52.1248     0.9652   54.00   <2e-16 ***
## log(lstat)  -12.4810     0.3946  -31.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.329 on 504 degrees of freedom
## Multiple R-squared:  0.6649, Adjusted R-squared:  0.6643
## F-statistic:  1000 on 1 and 504 DF,  p-value: < 2.2e-16
```

3.6.6 Qualitative Predictors

Loading the data called carseats

```r
data1 <- Carseats
kable(head(data1))
```

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

As we can see, from the above table ShelveLoc, Urban, US are qualitative variables.

Regression Model

```
lm.fit <- lm(Sales ~ . + Income:Advertising + Price:Age, data = data1)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Sales ~ . + Income:Advertising + Price:Age, data = data1)
##
## 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
```

A lot of the variables are not significant, thus we need to perform variable selection in order to determine the correct model.

```
contrasts(data1$ShelveLoc)
```

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

The above function gives the coding for the dummy variable ShelveLoc. We can find similar encoding for other categorical variables.