Lab 3 - Linear Regression

An Introduction to Statistical Learning

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

```r
summary(Boston)
```

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   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
```

1. Simple Linear Regression

Fitting the model

```r
lm.fit = lm(medv ~ lstat, data = Boston)
```

Calling `summary` we can see information about the fitted model:

- The minimum, maximum and quantile values for the residuals.
- The estimated values for the coefficients, as well as their standard error, and the T-statistic and p-value for the significance test.

- The residual standard error.
- The value of multiple R^2 and adjusted R^2.
- The value of the model F-statistic and its associated p-value. This statistic measures the relationship between the predictors and the response. When no relationship exists, the F-statistic is expected to be close to 1, whereas it would take values much greater than 1 when this relationship exists.

```
summary(lm.fit)
```

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

The fitted linear model has the following components:

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

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

We can print a 95% confidence interval for the coefficients using `confint` (the `level` argument defines the confidence level, defaults to 0.95):

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

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

Predictions

When `newdata` is not specified, predictions are done using the fitting (training) data.

```
lm.pred = predict(lm.fit)
```

If we want to pass a list of samples for which we want to predict values:

```
newdata = data.frame(lstat=c(2, 30))
lm.pred2 = predict(lm.fit, newdata = newdata)
lm.pred2
```

```
##        1        2
## 32.65374  6.05236
```

The `interval` argument for `predict` generates intervals for the predicted values. `confidence` returns the 95% *confidence* intervals for the prediction (only reducible error), while `prediction` returns *prediction* intervals considering both reducible and irreducible errors.

```
predict(lm.fit, newdata = newdata, interval = 'confidence')
```

```
##        fit       lwr       upr
## 1 32.65374 31.678068 33.629416
## 2  6.05236  4.625004  7.479716
```

```
predict(lm.fit, newdata = newdata, interval = 'prediction')
```

```
##        fit       lwr      upr
## 1 32.65374 20.402836 44.90465
## 2  6.05236 -6.242765 18.34749
```

Composing Features

Operations over the predictors can be done:

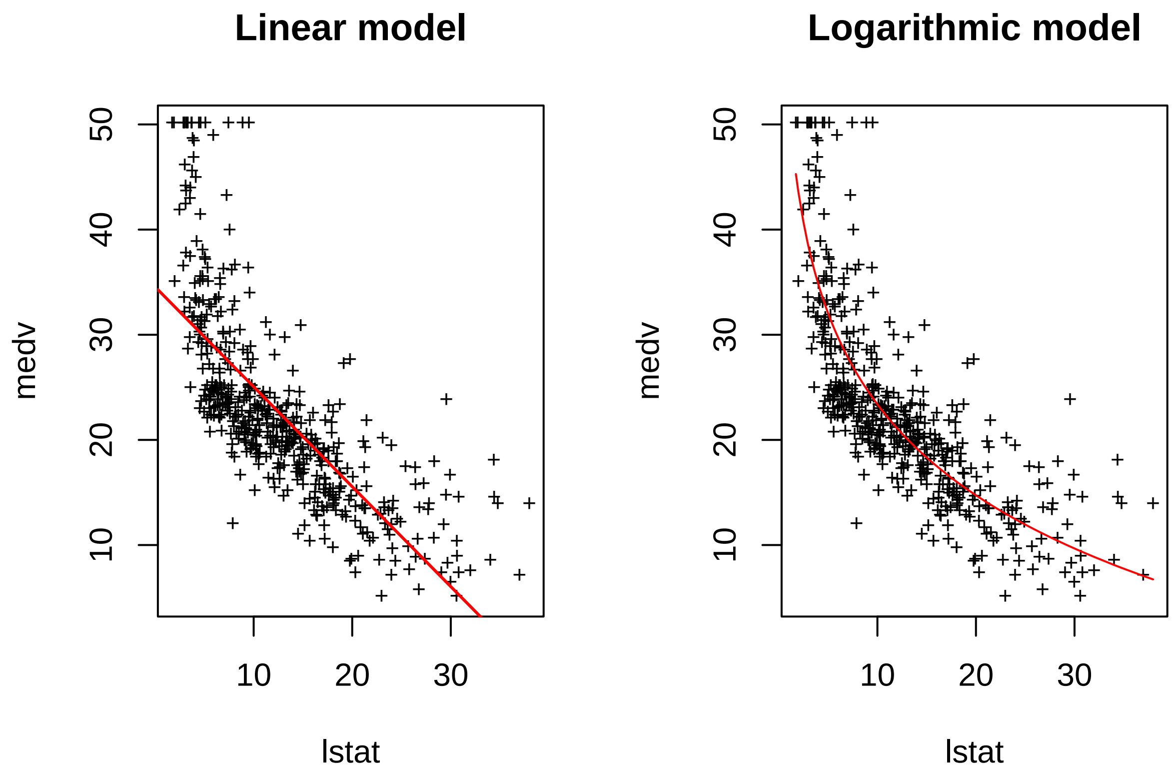
```
lm.fit2 = lm(medv ~ log(lstat), data=Boston)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ log(lstat), data = Boston)
##
## 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
```

Plotting the data

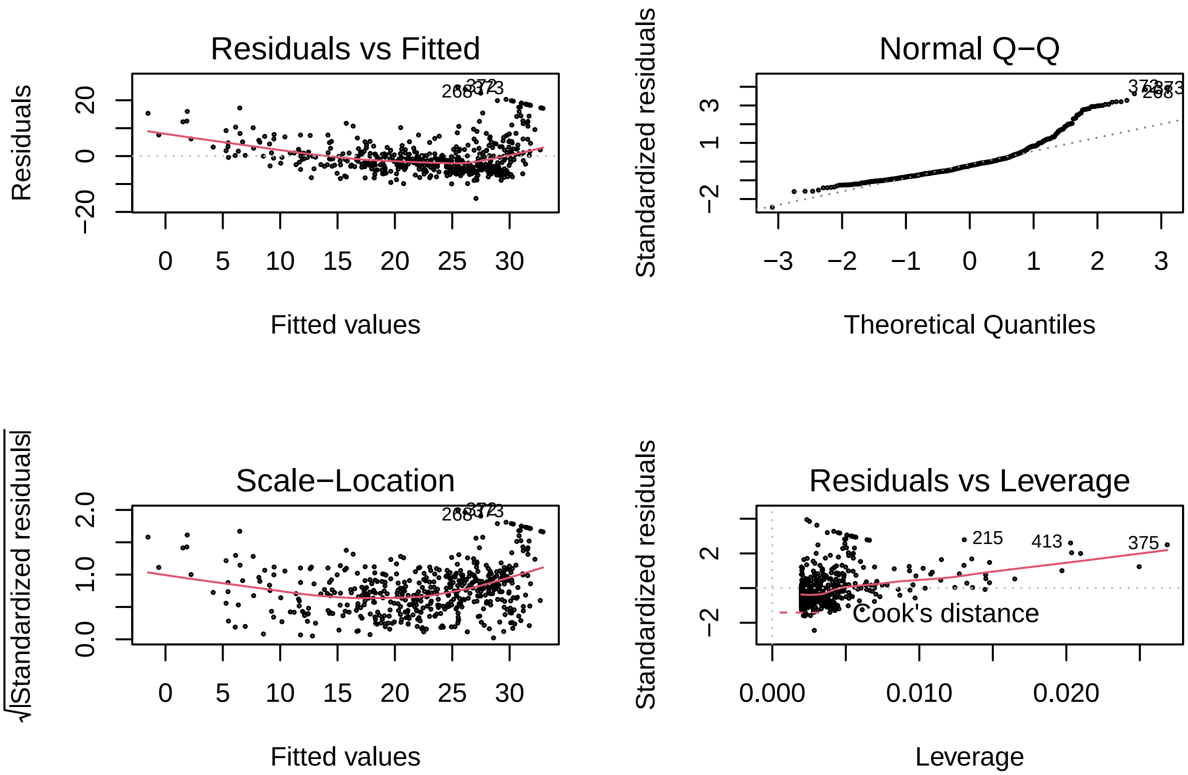
```
par(mfrow=c(1,2))
plot(lstat, medv, pch='+', cex=.75, title('Linear model'))
abline(lm.fit, col='red', lwd=1.5)

plot(lstat, medv, pch='+', cex=.75, title('Logarithmic model'))
lines(sort(lstat), fitted(lm.fit2)[order(lstat)], col='red')
```



The `lm` method comes with a pre-configured 2x2 plot:

```
par(mfrow=c(2,2))
plot(lm.fit, cex=.25)
```



- The first plot shows Residuals vs Fitted values. This can give an idea of the deviation from linearity by observing the residuals for the predicted values. It's equivalent to `plot(predict(lm.fit), residuals(lm.fit))`
- The second plot is a Normal Q-Q plot, that shows the difference between the model's residuals and a normal distribution, comparing the theoretical quantiles (the quantiles from a standard normal distribution).
- The third plot is obtained by standardizing the residuals from plot number one. Samples with an standardized residual greater than 3 could be considered an outlier. It's equivalent to `plot(predict(lm.fit), rstudent(lm.fit))`
- The fourth plot shows the residual vs the leverage of the sample points. It's equivalent to `plot(hatvalues(lm.fit), rstudent(lm.fit))`.

2. Multiple Linear Regression

Any kind of combination can be appliedo to features: `poly()`, `log()`, `sqrt()`...

```
lm.mult = lm(medv ~ poly(rm, 2)+ sqrt(lstat) + sqrt(rm), data = Boston)
summary(lm.mult)
```

```
##
## Call:
## lm(formula = medv ~ poly(rm, 2) + sqrt(lstat) + sqrt(rm), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -31.0319  -2.6935  -0.4258   2.0009  27.1910
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -341.4959   541.3489  -0.631   0.5284
## poly(rm, 2)1 -415.4564   680.0030  -0.611   0.5415
## poly(rm, 2)2   82.5833    38.1093   2.167   0.0307 *
## sqrt(lstat)    -5.5333     0.2812 -19.675   <2e-16 ***
## sqrt(rm)      152.9910   216.3760   0.707   0.4799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.609 on 501 degrees of freedom
## Multiple R-squared:  0.7509, Adjusted R-squared:  0.7489
## F-statistic: 377.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

Orthogonal and Non-Orthogonal Polynomials

`poly()` has a `raw` parameter that controls if orthogonal polynomials are created. It defaults to `FALSE`, creating orthogonal polynomials:

```
lm.poly = lm(wage ~ poly(age, 4, raw = FALSE), data = Wage)
coef(summary(lm.poly))
```

```
##                              Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)                 111.70361  0.7287409 153.283015 0.000000e+00
## poly(age, 4, raw = FALSE)1  447.06785 39.9147851  11.200558 1.484604e-28
## poly(age, 4, raw = FALSE)2 -478.31581 39.9147851 -11.983424 2.355831e-32
## poly(age, 4, raw = FALSE)3  125.52169 39.9147851   3.144742 1.678622e-03
## poly(age, 4, raw = FALSE)4  -77.91118 39.9147851  -1.951938 5.103865e-02
```

`poly()` returns a matrix whose columns are a basis of *orthogonal polynomials*, so each columns is a linear combination of the variables `age`, `age^2`, `age^3` and `age^4`.

If we set `raw=TRUE`:

```
lm.poly = lm(wage ~ poly(age, 4, raw = TRUE), data = Wage)
coef(summary(lm.poly))
```

```
##                                Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)               -1.841542e+02 6.004038e+01 -3.067172 0.0021802539
## poly(age, 4, raw = TRUE)1  2.124552e+01 5.886748e+00  3.609042 0.0003123618
## poly(age, 4, raw = TRUE)2 -5.638593e-01 2.061083e-01 -2.735743 0.0062606446
## poly(age, 4, raw = TRUE)3  6.810688e-03 3.065931e-03  2.221409 0.0263977518
## poly(age, 4, raw = TRUE)4 -3.203830e-05 1.641359e-05 -1.951938 0.0510386498
```

Now `poly()` returns `age`, `age^2`, `age^3` and `age^4` directly.

3. Qualitative Predictors

The Carseats dataset has both quantitative (numerical) and qualitative predictors:

```
attach(Carseats)
```

Fitting a linear model automatically generates dummy variables for the qualitative predictors:

```
lm.fit3 = lm(Sales ~ ., data=Carseats)
summary(lm.fit3)
```

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

```
## lm(formula = Sales ~ ., data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8692 -0.6908  0.0211  0.6636  3.4115
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.6606231  0.6034487   9.380  < 2e-16 ***
## CompPrice        0.0928153  0.0041477  22.378  < 2e-16 ***
## Income           0.0158028  0.0018451   8.565 2.58e-16 ***
## Advertising      0.1230951  0.0111237  11.066  < 2e-16 ***
## Population       0.0002079  0.0003705   0.561    0.575
## Price           -0.0953579  0.0026711 -35.700  < 2e-16 ***
## ShelveLocGood    4.8501827  0.1531100  31.678  < 2e-16 ***
## ShelveLocMedium  1.9567148  0.1261056  15.516  < 2e-16 ***
## Age             -0.0460452  0.0031817 -14.472  < 2e-16 ***
## Education       -0.0211018  0.0197205  -1.070    0.285
## UrbanYes         0.1228864  0.1129761   1.088    0.277
## USYes           -0.1840928  0.1498423  -1.229    0.220
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.019 on 388 degrees of freedom
## Multiple R-squared:  0.8734, Adjusted R-squared:  0.8698
## F-statistic: 243.4 on 11 and 388 DF,  p-value: < 2.2e-16
```

In this case, 3 qualitative predictors exist: `ShelveLoc`, `Urban` and `US`.

The model creates one dummy variable for each pair of classes in a predictor. The encoding can be shown using `contrasts()`:

```
contrasts(ShelveLoc)
```

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

For `ShelveLoc`, with 3 classes, 2 dummy variables are created: `ShelveLocGood`, encoded as `[1,0]` and `ShelveLocMedium`, encoded as `[0,1]`; the third class, corresponding to `Bad`, is the trivial encoding, `[0,0]`.