Chapter 3: Regression

- Simple linear regression
- Multiple linear regression
- Understanding regression results
- Non-linear effects in linear regression
- Regression tree

Regression - Simple Linear Regression

Let us first recall simple linear regression. Simple linear regression is a very straightforward simple linear approach for predicting a quantitative response Y on the basis of a single predictor variable X. Mathematically, we can write this linear relationship as

$$Y = \beta_0 + \beta_1 X + \epsilon.$$

Once we have used our training data to produce estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ for the model coefficients, we can predict future values of $\hat{y}$ on the basis of a particular x by computing

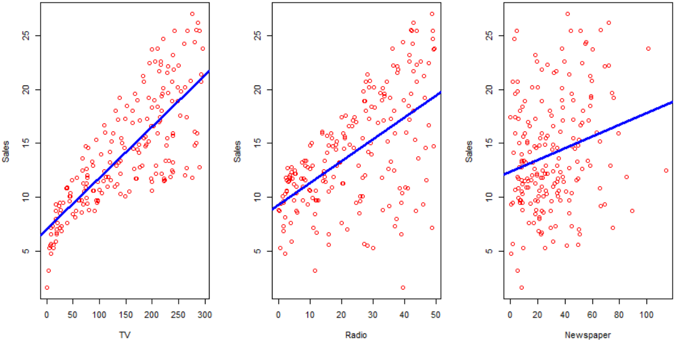
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

where $\hat{y}$ indicates a prediction of Y on the basis of $X = x$.

Regression - Simple Linear Regression

We will now plot *Sales* against *TV*, *Radio* and *Newspaper* respectively along with least squares regression line using the plot() and abline() functions.

```
par(mfrow = c(1, 3))
plot(Advertising$TV, Advertising$Sales, col = "red")
abline(lm(formula = Sales~TV , data = Advertising), lwd = 3, col = "
    blue") # Write other two plots by yourself.
...
```



Regression - Simple Linear Regression

We will start by using the lm() function to fit a simple linear regression model, with *TV* as the predictor and *Sales* as the response. The basic syntac is lm(y∼x, data), where y is the response, x is the predictor, and data is the data set in which these two variables are kept.

```
> library(ISLR)
> attach(Advertising)
> adSLR = lm(formula = Sales~TV)
 Error in eval(predvars, data, env) : object 'Sales' not found

> adSLR = lm(formula = Sales~TV, data = Advertising)
> attach(Advertising)
> adSLR = lm(formula = Sales~TV)
```

The command causes an error because R does not know where to find the variables *TV* and *Sales*. The next line tells that the variables are in Advertising. If we attach Advertising, R will recgonize the variables.

Regression - Simple Linear Regression

If we type adSLR, some basic information about the model is output.

```
> adSLR

Call:
lm(formula = Sales ~ TV)

Coefficients:
(Intercept)          TV
7.03259         0.04754
```

We can use the names() function in order to find out what other pieces of information are stored in adSLR.

```
> names(adSLR)
[1] "coefficients"  "residuals"     "effects"       "rank"
    "fitted.values" "assign"        "qr"            "df.residual"
[9] "xlevels"       "call"          "terms"         "model"
```

Regression - Simple Linear Regression

Detailed information about the model:

```
> summary(adSLR)

Call:
lm(formula = Sales ~ TV, data = Advertising)

Residuals:
Min        1Q    Median      3Q      Max
-8.3860  -1.9545  -0.1913   2.0671   7.2124

Coefficients:
Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)   7.032594   0.457843   15.36   <2e-16 ***
TV          0.047537   0.002691   17.67   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05  0.1  1

Residual standard error: 3.259 on 198 degrees of freedom
Multiple R-squared:  0.6119,  Adjusted R-squared:  0.6099
F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

Regression - Multiple Linear Regression

Simple linear regression is a useful approach for predicting a response on the basis of a single predictor variable. However, in practice we often have more than one predictor.

Instead of fitting a separate simple linear regression model for each predictor, a better approach is to extend the simple linear regression model so that it can directly accommodate multiple predictors. We can do this by giving each predictor a separate slope coefficient in a single model. In general, suppose that we have p distinct predictors. Then the multiple linear regression model takes the form

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon.$$

Once we have used our training data to produce estimates $\hat{\beta}_0, \ldots, \hat{\beta}_p$ for the model coefficients, we can predict future values of $\hat{y}$ on the basis of a particular x by computing

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p,$$

where $\hat{y}$ indicates a prediction of Y on the basis of $\boldsymbol{X} = (x_1, \ldots, x_p)$.

Regression - Multiple Linear Regression

```
> multipleLR = lm(formula = Sales~TV+Radio+Newspaper , data =
    Advertising)
> summary(multipleLR)
################################################################################
Call:
lm(formula = Sales ~ TV + Radio + Newspaper, data = Advertising)

Residuals:
    Min      1Q  Median      3Q     Max
-8.8277 -0.8908  0.2418  1.1893  2.8292

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.938889   0.311908   9.422   <2e-16 ***
TV           0.045765   0.001395  32.809   <2e-16 ***
Radio        0.188530   0.008611  21.893   <2e-16 ***
Newspaper   -0.001037   0.005871  -0.177     0.86
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05  0.1  1

Residual standard error: 1.686 on 196 degrees of freedom
Multiple R-squared:  0.8972,  Adjusted R-squared:  0.8956
F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

Regression - Multiple Linear Regression

```
> cor(Advertising[-1])

                  TV      Radio  Newspaper     Sales
TV        1.00000000 0.05480866 0.05664787 0.7822244
Radio     0.05480866 1.00000000 0.35410375 0.5762226
Newspaper 0.05664787 0.35410375 1.00000000 0.2282990
Sales     0.78222442 0.57622257 0.22829903 1.0000000
```

Since newspaper is not significant, we can refine the model using TV and radio only. With the model $sales = 2.9211 + 0.0458\,TV + 0.1880\,radio$, the predicted sales for spending 100 on TV and 20 on radio is 11.26, with the 95% confidence interval $[10.99, 11.53]$ and the 95% predictive interval $[7.93, 14.58]$.

```
> multipleLR = lm(formula = Sales~TV+Radio, data = Advertising)
> predict(multipleLR, data.frame(TV = c(100), Radio = c(20)),
    interval = "confidence")
       fit      lwr      upr
1 11.25647 10.98525 11.52768
> predict(multipleLR, data.frame(TV = c(100), Radio = c(20)),
    interval = "predict")
       fit      lwr      upr
1 11.25647 7.929616 14.58332
```

Regression - Interaction Terms

Linear regression models are linear in coefficients $\beta_0, \beta_1, \ldots, \beta_p$, while regressors $X_1, \ldots, X_p$ can be replaced by any known functions of them.

```
> nonLR = lm(formula = Sales~TV*Radio, data = Advertising)
> coefficients(summary(nonLR))

               Estimate   Std. Error   t value     Pr(>|t|)
(Intercept) 6.750220203 0.2478713699 27.232755 1.541461e-68
TV          0.019101074 0.0015041455 12.698953 2.363605e-27
Radio       0.028860340 0.0089052729  3.240815 1.400461e-03
TV:Radio    0.001086495 0.0000524204 20.726564 2.757681e-51
```

In this example, the p-values associated with TV, radio, and the interaction term all are statistically significant, and so it is obvious that all three variables should be included in the model. The hierarchical principle states that if we include an interaction in a model, we hierarchical should also include the main effects, even if the p-values associated with principle their coefficients are not significant.

Regression - Polynomial Regression

The data set Auto contains various indices for 387 cars. Let us consider the relationship between mpg (gas mileage in miles power gallon) versus horsepower. Looking at scatter plots, the relationship does not appear to be linear. We fit polynomial regression:

$$mpg = \beta_0 + \beta_1 hpower + \cdots + \beta_p hopower^p + \epsilon, \quad \text{for } p \in \mathbb{N}.$$

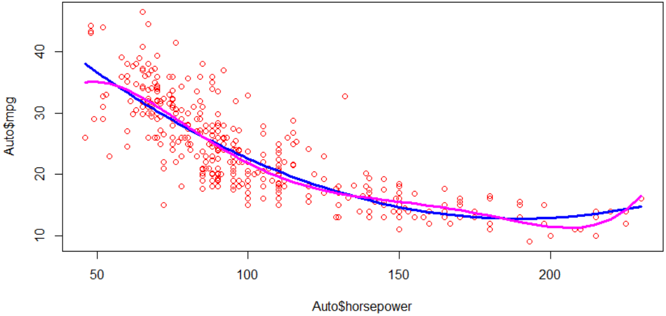
The results for $p = 2$ are listed below

```
> polynomialLR = lm(formula = Auto$mpg~Auto$horsepower+I(Auto$
    horsepower^2), data = Auto)
> coefficients(summary(polynomialLR))
Estimate                 Std.     Error     t value    Pr(>|t|)
(Intercept)              56.9001  1.80042   31.60367   1.74091e-109
Auto$horsepower          -0.4662  0.03112   -14.9781   2.2829e-40
I(Auto$horsepower^2)     0.00123  0.000122  10.08009   2.1963e-21
```

The function I() is needed since the ^2 has a special meaning in a formula.

Regression - Polynomial Regression

```
> plot(Auto$horsepower, Auto$mpg, col = "red")
> lines(sort(Auto$horsepower), fitted(polynomialLR)[order(Auto$
    horsepower)], col='blue', type='l', lwd = 3)
> polynomialLR = lm(formula = Auto$mpg~poly(Auto$horsepower, 5),
    data = Auto)
> lines(sort(Auto$horsepower), fitted(polynomialLR)[order(Auto$
    horsepower)], col = 6, type='l', lwd = 3)
```



Regression - Function 'sort' & 'order'

sort(): sort a vector or factor (partially) into ascending or descending order. order(): return a permutation which rearranges its first argument into ascending or descending order. Use ?sort and ?order to learn about those two functions with more examples.

```
> set.seed(1)
> a = sample(c(1:5))
> a
 [1] 2 5 4 3 1
> sort(a)
 [1] 1 2 3 4 5
> order(a)
 [1] 5 1 4 3 2
```

Question: What is the output of a[order(a)] ?

Regression - Regression Tree

Here we fit a regression tree to the Boston data set (part of MASS package). First, we create a training set, and fit the tree to the training data. In the context of a regression tree, the deviance is simply the sum of squared errors for the tree.

```
> library(MASS)
> library(tree)
> set.seed(1)
> train = sample(1:nrow(Boston), nrow(Boston)/2)
> tree.boston = tree(medv~., Boston, subset = train)
> summary(tree.boston)

 Regression tree:
 tree(formula = medv ~ ., data = Boston, subset = train)
 Variables actually used in tree construction:
 [1] "lstat" "rm"    "dis"
 Number of terminal nodes:  8
 Residual mean deviance:  12.65 = 3099 / 245
 Distribution of residuals:
 Min.      1st Qu.    Median     Mean     3rd Qu.    Max.
 -14.10000  -2.04200  -0.05357  0.00000  1.96000  12.60000
```

Regression - Regression Tree

Now we use the cv.tree() to see whether pruning the tree will improve performance. In this case, the most complex tree is selected by cross-validation.

```
> cv.boston = cv.tree(tree.boston)
> plot(cv.boston$size, cv.boston$dev, type = 'b')
```

However, if we wish to prune the tree, we could do so as follows, using prune.tree() function.

```
> prune.boston = prune.tree(tree.boston, best = 5)
> plot(prune.boston)
> text(prune.boston, pretty = 0)
```

Regression - Regression Tree

In keeping with the cross-validaion results, we use the unpruned tree to make predictions on the test set.

```
> yhat = predict(tree.boston, newdata = Boston[-train,])
> boston.test = Boston[-train, "medv"]
> plot(yhat, boston.test)
> abline(0,1)
> mean((yhat-boston.test)^2)
[1] 25.04559
```