Predictive Analytics Linear and Nonlinear Models Using Auto data

Chris Schmidt

2019

Linear and Nonlinear Models using the Auto Data from the ISLR Package

In this project we are building and analyzing linear and nonlinear models using the Auto data set found in the ISLR package in R.

We are looking to answer a few questions:

- Is at least one of the predictors useful in predicting the response?
- What is the best combination of predictors or is there one dominant predictor that is best?
- How well does the model specified predict the response?
- What response should be predicted and how accurate is the prediction given a set of attributes at predictor values?
- Do we have a linear or non-linear relationship?

With linear regression we want to know if there is a relationship between the response and predictors.

The model form given response Y and predictors X_j is

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

for predictors X_j for $j = 1, ..., p$ and where β_j is the association between the jth variable and the response Y.

The interpretation of β_j is the **average** effect on Y of a one unit increase in X_j, **holding all other predictors fixed**.

Because we don't know the regression coefficients we need to estimate them so that given the estimates $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p$ we can make predictions using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ... + \hat{\beta}_p x_p$$

We find the parameter estimates using least square to solve for

$$\begin{aligned} RSS &= \sum_{i=1}^{n} (y_i - \hat{y})^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - ... - \hat{\beta}_p x_{ip})^2 \end{aligned}$$

Using the null hypothesis

$$H_0 = \beta_1 = \beta_2 = ... = \beta_p = 0$$

for predictors $1, ..., p$ we can compare to the alternative hypothesis

$$H_A : \text{ at least one } \beta_j \text{ is non-zero.}$$

by performing this hypothesis test using the **F-statistic**

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

where $TSS = \sum(y_i - \bar{y})^2$ and $RSS = \sum(y_i - \hat{y}_i)^2$.

Where the linear model assumptions are correct we can show that

$$E[RSS/(n-p-1)] = \sigma^2$$

and, if the null hypothesis is true,

$$E[(TSS - RSS)/p] = \sigma^2$$

This results in an ability to use the **F-statistic** to disern the strength of the relationship between the response and the predictors. If there is no relationship between the two, the **F-statistic** will have a value close to 1 and if the alternative hypothesis, H_A is true, then $E[(TSS - RSS)/p] > \sigma^2$ where **F** will be larger than 1.

Set Up Your Project and Load Libraries

We need the `ISLR` and `mgcv` packages.

The Auto data

This data set contains 392 observations with 9 attributes as detailed below.

- mpg miles per gallon
- cylinders Number of cylinders between 4 and 8
- displacement Engine displacement (cu. inches)
- horsepower Engine horsepower
- weight Vehicle weight (lbs.)
- acceleration
- Time to accelerate from 0 to 60 mph (sec.)
- year Model year (modulo 100)
- origin Origin of car (1. American, 2. European, 3. Japanese)
- name Vehicle name

We examine the structure of the data set by using the **str()** function with the data set as the argument.

```
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
```

The original data contained 408 observations but 16 observations with missing values were removed. Using the head() function on the data set we can look at the first 6 rows of data and the column headers.

```
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                       name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition.

Building our Predictive Models

Note that 'origin' is a number in the Auto data set. We want it to be a categorical variable so an important step is to make the transformation to ensure that the variable 'origin' is treated properly for our needs.

We will use the **as.factor()** function to transform the 'origin' variable to a factor and also to split our data set into a training set and a testing set using a 70/30 split of the data. We use the **set.seed()** function to produce the same split of the date set each time we run the test/train split for consistency purposes.

After transforming the 'origin' attribute to a factor type, the Auto data set is renamed Auto_S and we name the training data set 'auto' in lower case.

```
class(Auto$origin)
```

```
## [1] "numeric"
```

```
set.seed(1)
n=dim(Auto)[1]
train <- sample(n, n*0.7)

Auto_S <- Auto
Auto_S$origin <- as.factor(Auto$origin)
auto <- Auto_S[train, ]
```

Using the GGally package and the **ggpairs()** function we can look at a pairs plot to develop a visual understanding of the relationships and influences the attributes have on each other.

```
library(GGally)
ggpairs(Auto_S, columns = c(2:7, 1,8), upper = list(continuous = ggally_density, combo ='box'),   lower
```

We can look at the first few rows of our transformed data set and the structure, **Auto_S**.

```
head(Auto_S)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
```

```
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                         name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
str(Auto_S)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
```

Building the first model

We will call our first linear model **m1** with 'mpg' as our response variable and using 'cylinders', 'displacement', 'horsepower', 'weight', and 'acceleration' as our predictors for the miles per gallon of the vehicles in the data.

```
m1 <- lm(mpg ~ cylinders + displacement
         + horsepower + weight + acceleration, data = Auto_S)
```

The **summary()** function generates information about the residuals, the p-values and standard errors of the predictors and model strength as indicated by the **F-statistic** and the R^2**R** and adjusted R^2 values.

For the model **m1** we have an adjusted R^2 value of 0.7039 which indicates that the model explains roughly 70% of the variance in the response variable. The **F-statistic** value of 186.9 on 5 predictors lets us reject H_0 indicating that there is a relationship between the predictors and reponse. The p-values for each predictor test the null hypothesis H_0 that the coefficient has no effect on the response variable. If the p-value is below the significance level (the default is 5%) then we reject the null hypothesis and conclude that the predictor contributes to changes in the response variable.

The model summary shows that horsepower and weight are significant contributers to changes in the response variable for this combination of the predictor variable.

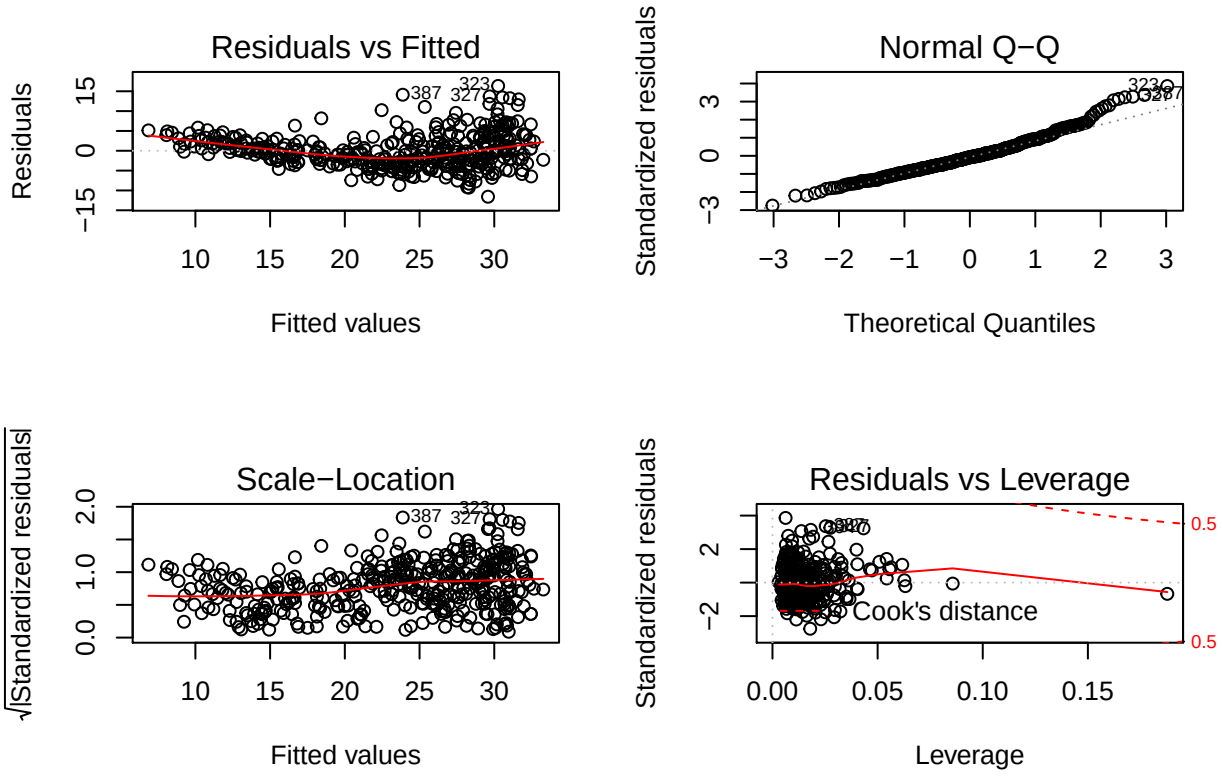
```
summary(m1)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration, data = Auto_S)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5816  -2.8618  -0.3404   2.2438  16.3416
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.626e+01  2.669e+00  17.331   <2e-16 ***
## cylinders    -3.979e-01  4.105e-01  -0.969   0.3330
## displacement -8.313e-05  9.072e-03  -0.009   0.9927
## horsepower   -4.526e-02  1.666e-02  -2.716   0.0069 **
## weight       -5.187e-03  8.167e-04  -6.351    6e-10 ***
## acceleration -2.910e-02  1.258e-01  -0.231   0.8171
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.247 on 386 degrees of freedom
## Multiple R-squared:  0.7077, Adjusted R-squared:  0.7039
## F-statistic: 186.9 on 5 and 386 DF,  p-value: < 2.2e-16
```

Plotting the model generates a view of the residual versus fitted values, a Quantile-Quantile plot for the normality assumption of the residuals, a scale-location plot for standardized versus fitted values and a Cook's distance metric for residuals verus leverage.

```
par(mfrow=c(2,2))
plot(m1)
```



Polynomial regression models with interraction effect between `horsepower` and `origin`

If we have a non-linear relationship between the response and predictors we can extend the linear model using polynomial regression by adding interaction effects. Interaction effects can be modeled by using the product of two predictors or by using a polynomial applied to a predictor.

For example, if we include a third predictor in a multiple linear regression model with two predictors where the product of X_1 and X_2 produces the interaction term X_1X_2 as our third predictor, our model would look like

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$$

which can be represented as

$$\begin{aligned} Y &= \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \beta_2 X_2 + \epsilon \\ &= \beta_0 + \tilde{\beta}_1 X_1 + \beta_2 X_2 + \epsilon \end{aligned}$$

where $\tilde{\beta}_1 = \beta_1 + \beta_3 X_2$.

We can also replace the standard linear model with a polynomial function where we have a nonlinear relationship between the response and the predictors using

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_d x_i^d + \epsilon$$

where d is the degree of the polynomial. The polynomial can be applied to specific attributes

Below we build out three models $m2, m3, m4$ to look at several combinations of interaction effects on the training data.

```
m2 <- lm(mpg ~ cylinders + poly(displacement,5) + weight
         + acceleration + poly(year,2) + I(displacement*weight)
         + weight:origin, data = Auto_S)

test_pre <- predict(m2, newdata = Auto_S[-train, ])
mean(abs(test_pre - Auto_S[-train, ]$mpg))
```

```
## [1] 2.030879
```

```
m3 <- lm(mpg ~ cylinders + poly(displacement, 2) + horsepower + weight
         + acceleration + poly(year, 2) + I(horsepower*weight) + horsepower:origin
         + I(horsepower * displacement), data = Auto_S)

test_pre <- predict(m3, newdata = Auto_S[-train, ])
mean(abs(test_pre-Auto_S[-train, ]$mpg))
```

```
## [1] 1.859873
```

```
mean(abs(residuals(m3)))
```

```
## [1] 1.971373
```

```
m4 <- lm(mpg ~ cylinders + poly(displacement, 2) + weight + acceleration
         + poly(year, 2) + I(displacement * weight) + I(horsepower * displacement)
         + horsepower:origin, data = Auto_S)

test_pre <- predict(m4, newdata = Auto_S[-train, ])
mean(abs(test_pre-Auto_S[-train, ]$mpg))
```

```
## [1] 1.85163
```

```
mean(abs(residuals(m4)))
```

```
## [1] 1.944851
```

```r
cat('The testing mean absolute error of model m5 is', mean(abs(test_pre-Auto_S[-train, ]$mpg)), '.\n')
```

```
## The testing mean absolute error of model m5 is 1.85163 .
```

```r
summary(m2)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + poly(displacement, 5) + weight +
##     acceleration + poly(year, 2) + I(displacement * weight) +
##     weight:origin, data = Auto_S)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1817 -1.6119  0.0932  1.5483 12.5476
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.789e+01  2.850e+00  13.294  < 2e-16 ***
## cylinders                  1.570e-01  3.732e-01   0.421 0.674261
## poly(displacement, 5)1    -1.749e+02  3.154e+01  -5.546 5.51e-08 ***
## poly(displacement, 5)2    -1.086e+01  7.999e+00  -1.357 0.175491
## poly(displacement, 5)3    -8.517e-01  3.750e+00  -0.227 0.820467
## poly(displacement, 5)4    -5.197e-01  3.203e+00  -0.162 0.871175
## poly(displacement, 5)5     3.015e+00  2.939e+00   1.026 0.305571
## weight                    -1.274e-02  1.385e-03  -9.201  < 2e-16 ***
## acceleration               2.423e-01  7.163e-02   3.383 0.000792 ***
## poly(year, 2)1             6.033e+01  3.249e+00  18.570  < 2e-16 ***
## poly(year, 2)2             1.697e+01  3.033e+00   5.596 4.23e-08 ***
## I(displacement * weight)   2.776e-05  4.805e-06   5.777 1.59e-08 ***
## weight:origin2             7.112e-04  2.249e-04   3.162 0.001691 **
## weight:origin3             4.313e-04  2.228e-04   1.935 0.053695 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.864 on 378 degrees of freedom
## Multiple R-squared:  0.8698, Adjusted R-squared:  0.8653
## F-statistic: 194.3 on 13 and 378 DF,  p-value: < 2.2e-16
```

```r
summary(m3)
```

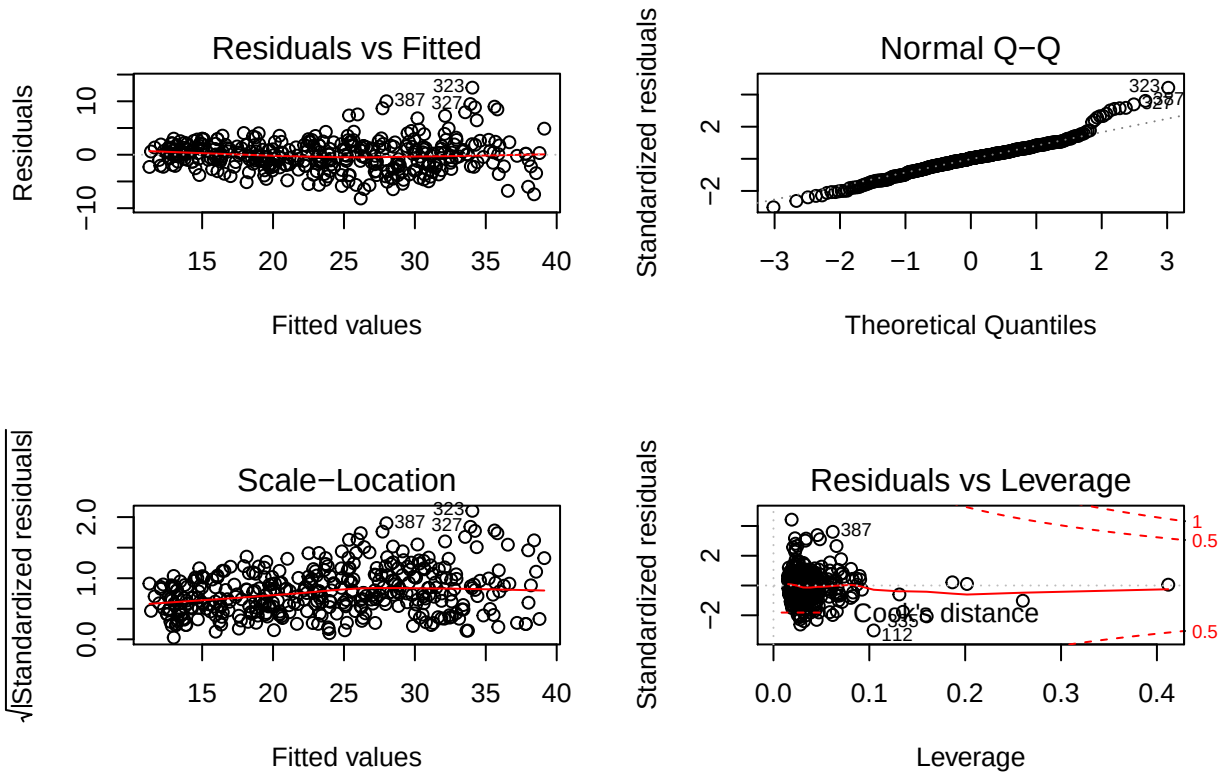
```
##
## Call:
## lm(formula = mpg ~ cylinders + poly(displacement, 2) + horsepower +
##     weight + acceleration + poly(year, 2) + I(horsepower * weight) +
##     horsepower:origin + I(horsepower * displacement), data = Auto_S)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.0043 -1.4363 -0.0195  1.2564 11.8455
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.004e+01  5.005e+00   9.999  < 2e-16 ***
## cylinders                     6.579e-01  3.092e-01   2.128  0.03398 *
```

```
## poly(displacement, 2)1       -8.264e+01  3.948e+01  -2.093  0.03700 *
## poly(displacement, 2)2        4.902e+00  8.080e+00   0.607  0.54442
## horsepower                   -2.350e-01  3.196e-02  -7.353 1.20e-12 ***
## weight                       -6.398e-03  1.580e-03  -4.049 6.23e-05 ***
## acceleration                 -1.753e-01  9.173e-02  -1.911  0.05682 .
## poly(year, 2)1                5.461e+01  3.234e+00  16.887  < 2e-16 ***
## poly(year, 2)2                1.736e+01  2.956e+00   5.872 9.42e-09 ***
## I(horsepower * weight)        2.788e-05  1.026e-05   2.718  0.00688 **
## I(horsepower * displacement)  2.611e-04  1.284e-04   2.034  0.04262 *
## horsepower:origin2            1.296e-02  6.359e-03   2.038  0.04222 *
## horsepower:origin3            1.481e-02  6.213e-03   2.383  0.01766 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.784 on 379 degrees of freedom
## Multiple R-squared:  0.8767, Adjusted R-squared:  0.8727
## F-statistic: 224.5 on 12 and 379 DF,  p-value: < 2.2e-16
```

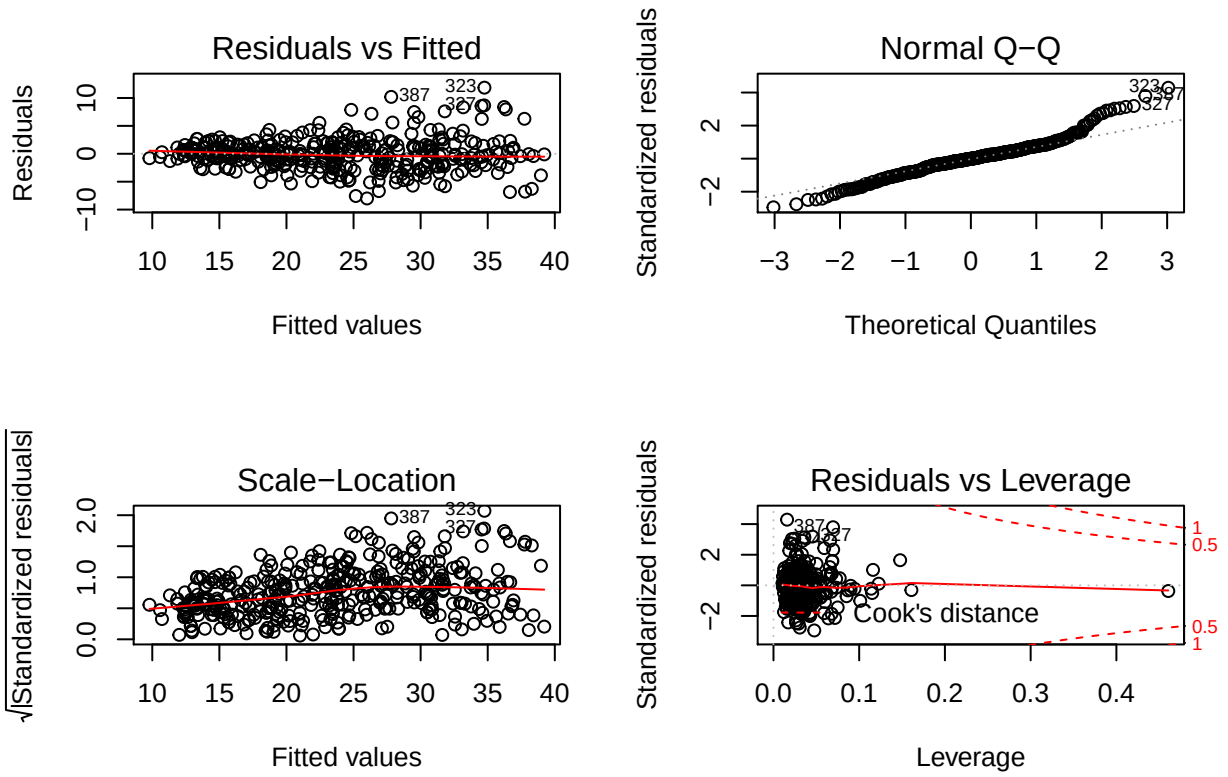
```
summary(m4)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + poly(displacement, 2) + weight +
##     acceleration + poly(year, 2) + I(displacement * weight) +
##     I(horsepower * displacement) + horsepower:origin, data = Auto_S)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8927 -1.5436 -0.0255  1.2852 11.9546
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   4.066e+01  2.687e+00  15.130  < 2e-16 ***
## cylinders                     5.574e-01  3.087e-01   1.806 0.071774 .
## poly(displacement, 2)1       -2.258e+02  3.492e+01  -6.467 3.08e-10 ***
## poly(displacement, 2)2       -1.650e+01  1.007e+01  -1.638 0.102332
## weight                       -7.211e-03  1.469e-03  -4.910 1.36e-06 ***
## acceleration                 -1.316e-01  9.294e-02  -1.416 0.157540
## poly(year, 2)1                5.525e+01  3.220e+00  17.158  < 2e-16 ***
## poly(year, 2)2                1.799e+01  2.939e+00   6.120 2.34e-09 ***
## I(displacement * weight)      1.616e-05  4.510e-06   3.582 0.000385 ***
## I(horsepower * displacement)  4.278e-04  9.962e-05   4.295 2.23e-05 ***
## horsepower:origin1           -1.737e-01  3.054e-02  -5.686 2.60e-08 ***
## horsepower:origin2           -1.575e-01  3.019e-02  -5.218 2.98e-07 ***
## horsepower:origin3           -1.579e-01  2.918e-02  -5.412 1.11e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.765 on 379 degrees of freedom
## Multiple R-squared:  0.8784, Adjusted R-squared:  0.8745
## F-statistic: 228.1 on 12 and 379 DF,  p-value: < 2.2e-16
```

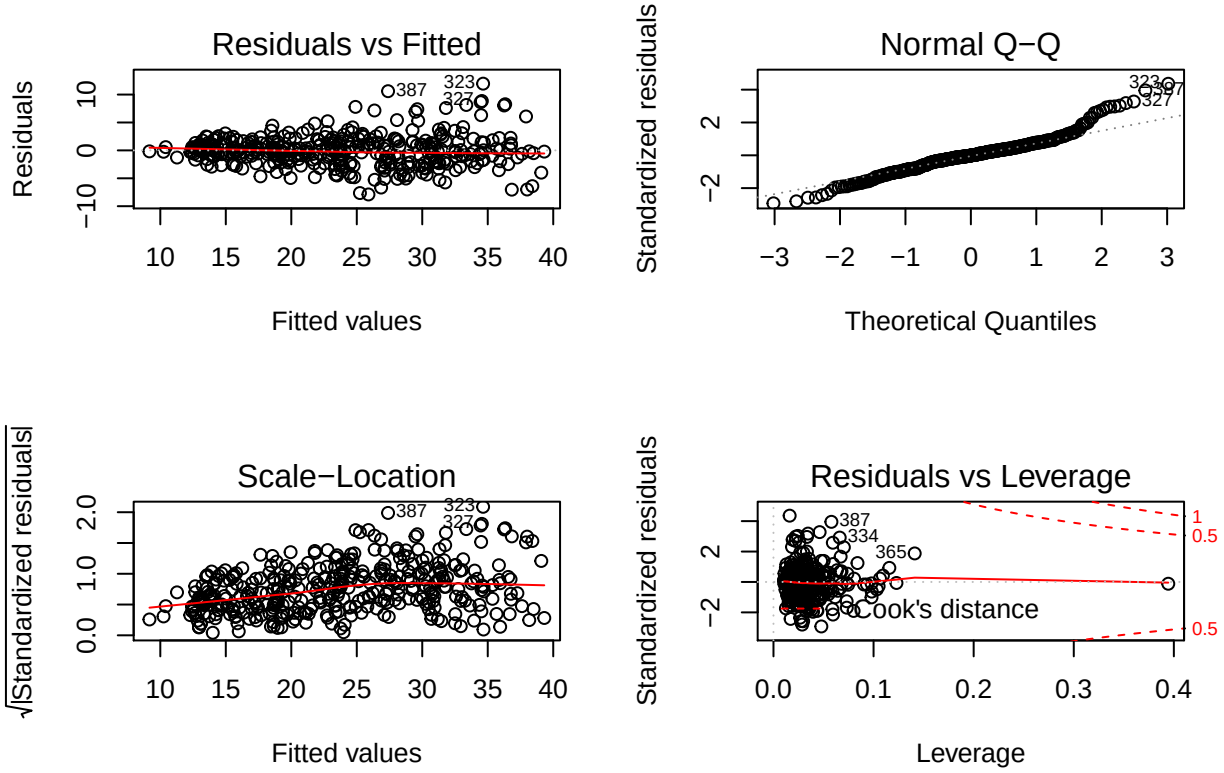
```
par(mfrow=c(2,2))
plot(m2)
```




```
par(mfrow=c(2,2))
plot(m3)
```




```
par(mfrow=c(2,2))
plot(m4)
```



Can we develop a model using `lm()` with lower test mean absolute error than m4?

Yes. By adding an interaction term I(displacement*acceleration) to the model, m4, the testing mean absolute error dropped to 1.838677.

The MAE (mean absolute error) is a measure of the errors between between the the target and the predicted variables. It is the sum of absolute differences between the two and is represented by

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$

We find the MAE by using the **mean(), and abs()** functions on the difference between the results of running the **m4T** model created below on the test data (the holdout data) and subtracting the training data results for the model as shown below.

```
m4T <- lm(mpg ~ cylinders + poly(displacement,2) + weight + acceleration
          + poly(year,2) + I(displacement*weight) + I(horsepower*displacement)
          + horsepower:origin + I(displacement * acceleration), data = Auto_S)

test_pre <- predict(m4T, newdata = Auto_S[-train, ])

mean(abs(test_pre - Auto_S[-train, ]$mpg))
```

```
## [1] 1.838677
```

```
cat('The testing mean absolute error of model m4T is', mean(abs(test_pre-Auto_S[-train, ]$mpg)), '.\n')
```

```
## The testing mean absolute error of model m4T is 1.838677 .
```

Spline Regression models

This is a cursory introduction to spline repgression modeling using the **gam()** function to build generalized additive models

If we fit piece-wise low degree polynomials over dfference regions of our predictors X_i where the coefficients $\beta_0, \beta_1, \beta_2, ..., \beta_p$ vary across X we create a piece-wise polynomial regression. The points where the coeffients change are called **knots**.

A piece-wise polynomial for a cubic regression would have the form

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon$$

and a piece-wise cubic polynomial with a single knot at a point c has the form

$$y_i = \begin{cases} \beta_0 + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon \text{ if } x_i < c; \\ \beta_0 + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon \text{ if } x_i \geq c \end{cases}$$

The big idea behind regression splines is the flexibility introduced by allowing the introduction of more knots and leaving the degree unchanged which generally produces more stability.

Below we create three models using this approach and generate the **summary()** output.

```
set.seed(1)
n <- dim(Auto)[1]
train <- sample(n, n*0.7)

Auto_S <- Auto
Auto_S$origin <- as.factor(Auto$origin)
auto=Auto_S[train, ]

m1_sr <- gam(mpg ~ cylinders + s(year) + s(acceleration) + s(weight)
             + s(horsepower) + ti(horsepower, weight, by=origin)
             + ti(acceleration, horsepower) + ti(acceleration, weight), data = auto)

test_pre <- predict(m1_sr, newdata = Auto_S[-train, ])
mean(abs(test_pre - Auto_S[-train, ]$mpg))
```

```
## [1] 1.802063
```

```
mean(abs(residuals(m1_sr)))
```

```
## [1] 1.694704
```

```
m2_sr <- gam(mpg ~ cylinders + s(year) + acceleration + s(weight)
             + s(horsepower) + ti(weight, horsepower, by = origin), data=auto)

test_pre <- predict(m2_sr, newdata=Auto_S[-train, ])
mean(abs(test_pre - Auto_S[-train, ]$mpg))
```

```
## [1] 1.7968
```

```
m3_sr <- gam(mpg ~ cylinders + s(year) + acceleration + s(weight)
             + s(horsepower, by =origin) + ti(acceleration, horsepower)
             +  ti(weight, horsepower, by = origin), data = auto)

test_pre <- predict(m3_sr, newdata = Auto_S[-train, ])
mean(abs(test_pre - Auto_S[-train, ]$mpg))
```

```
## [1] 1.777841
```

```
mean(abs(residuals(m3_sr)))
```

```
## [1] 1.740308
```

```
summary(m1_sr)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ cylinders + s(year) + s(acceleration) + s(weight) + s(horsepower) +
##     ti(horsepower, weight, by = origin) + ti(acceleration, horsepower) +
##     ti(acceleration, weight)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.5158     1.8605  10.490   <2e-16 ***
## cylinders     0.2517     0.3213   0.783    0.434
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                                edf Ref.df      F  p-value
## s(year)                      8.185  8.802 33.139  < 2e-16 ***
## s(acceleration)              1.000  1.000  4.224 0.040965 *
## s(weight)                    1.000  1.000 31.367  < 2e-16 ***
## s(horsepower)                2.553  3.280  2.275 0.078847 .
## ti(horsepower,weight):origin1 7.479  8.621  3.629 0.000849 ***
## ti(horsepower,weight):origin2 6.879  8.168  4.182 9.97e-05 ***
## ti(horsepower,weight):origin3 2.315  2.982  2.622 0.056668 .
## ti(acceleration,horsepower)   5.847  7.435  1.148 0.324642
## ti(acceleration,weight)       2.403  2.857  1.235 0.254586
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.902   Deviance explained = 91.6%
## GCV = 7.2721  Scale est. = 6.2195    n = 274
```

```
summary(m2_sr)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ cylinders + s(year) + acceleration + s(weight) + s(horsepower) +
##     ti(weight, horsepower, by = origin)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25.30895    2.33477  10.840   <2e-16 ***
## cylinders     0.08627    0.26951   0.320   0.7492
## acceleration -0.22020    0.09945  -2.214   0.0277 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                                edf Ref.df      F  p-value
## s(year)                      8.505  8.924 33.918  < 2e-16 ***
## s(weight)                    2.573  3.412 22.654  < 2e-16 ***
## s(horsepower)                1.000  1.000 12.180 0.000572 ***
## ti(weight,horsepower):origin1 1.000  1.000  8.000 0.005058 **
## ti(weight,horsepower):origin2 7.189  8.536  7.323  < 2e-16 ***
## ti(weight,horsepower):origin3 2.099  2.688  2.698 0.062776 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.898   Deviance explained = 90.7%
## GCV =  7.097  Scale est. = 6.44      n = 274
```

```
summary(m3_sr)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ cylinders + s(year) + acceleration + s(weight) + s(horsepower,
##     by = origin) + ti(acceleration, horsepower) + ti(weight,
##     horsepower, by = origin)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   25.0884     2.5829   9.713   <2e-16 ***
## cylinders      0.1929     0.2993   0.645   0.5198
## acceleration  -0.2521     0.1023  -2.464   0.0144 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                                     edf Ref.df      F  p-value
## s(year)                       8.318e+00  8.864 31.192  < 2e-16 ***
## s(weight)                     2.668e+00  3.541 20.347  < 2e-16 ***
## s(horsepower):origin1         1.000e+00  1.000 15.400 0.000114 ***
## s(horsepower):origin2         3.557e+00  4.115  4.854 0.000799 ***
## s(horsepower):origin3         5.485e+00  5.945  4.514 0.000314 ***
## ti(acceleration,horsepower)   2.134e+00  2.744  1.003 0.489163
## ti(weight,horsepower):origin1 1.000e+00  1.000  6.678 0.010348 *
## ti(weight,horsepower):origin2 3.430e+00  4.435  5.528 0.000189 ***
## ti(weight,horsepower):origin3 1.409e-10 15.000  0.000 0.960083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.902   Deviance explained = 91.2%
## GCV = 6.9997  Scale est. = 6.2182    n = 274
```

Develop a model using gam() with lower test mean absolute error than m1_sr.

m1_sr <- gam(mpg ~ cylinders + s(year) + s(acceleration) + s(weight) + s(horsepower) + ti(horsepower, weight, by=origin) + ti(acceleration, horsepower) + ti(acceleration, weight), data = auto)

```
mT_sr <- gam(mpg ~ cylinders + s(year) + acceleration + s(weight)
             + s(horsepower, by = origin) + ti(acceleration, horsepower)
             +  ti(weight, horsepower, by = origin)
             + ti(acceleration, displacement), data = auto)

test_pre <- predict(mT_sr, newdata = Auto_S[-train, ])


mean(abs(test_pre - Auto_S[-train, ]$mpg))
```

```
## [1] 1.792112
```

The mT_sr model which added ti(acceleration, displacement) to the mix of predictors has a lower test mean absolute error at 1.792112 than the m1_sr model which was 1.802063.