STAT350 Tutorial 5

Dylan Maciel

22/09/2020

Last week we covered visual checks of model assumptions for multiple linear regression. This week we will go over detection of outliers and a test for lack of fit, then we'll get into transforming the data to correct a model that fails to meet one or more assumptions.

Here's the model from last week:

```
##
## Call:
## lm(formula = mpg ~ wt + hp + disp, data = auto_mpg)
##
## Coefficients:
## (Intercept)           wt           hp         disp
##   44.855936    -0.005352    -0.041674    -0.005769
```

Detection of Outliers

Recall, an outlier is an extreme observation in the data that have the potential to greatly impact the regression model. In the residual plots covered last week there were a couple ways to check for potential outliers; residuals with large absolute values and points with high leverage should be further investigated. Keeping in mind that an outlier is not a point that will just be deleted from the data set – their presence can indicate more than experimental error.

With our linear regression model on the automotive fuel efficiency data we saw no extreme residuals or points with high leverage. If one or two such points did exist within the dataset we could proceed by fitting a regression model to the data with those points removed. In doing this we are looking for any dramatic changes in the models coefficient estimates, their estimated standard errors, and the summary statistics for the model which would suggest that the should be a cause for concern. Example 4.7 in your textbook provides a good illustration of the process you should follow.

Test for Lack of Fit

Assuming that all other model assumptions are met, we can use the F-test for lack of fit of the regression model to see if the linear relationship requirement is met. This boils down to making sure that the error in the model is a result of the error in the data and not due to the model being in effective in capturing the true nature of the relationship between the dependent and independent variables. To do this we'll need a model independent measure of σ^2, so the test requires replicate observations in the data.

Our hypotheses are:

$$\text{H}_0: \text{ There is no lack of fit in the model.}$$

vs.

$$\text{H}_A: \text{ There is lack of fit in the model.}$$

The test statistic is

$$F_0 = \frac{MS_{LOF}}{MS_{PE}}.$$

For some intuition we'll look at the expected values of these two quantities. First, the denominator contains the mean squared pure error,

$$E[MS_{PE}] = \sigma^2.$$

And, the numerator contains the mean square error due to lack of fit

$$E[MS_{LOF}] = \sigma^2 + \frac{\sum_{i=1}^{m} n_i(E[y_i] - \beta_0 - \beta_1 x_i)^2}{m-2}.$$

So, if there is no lack of fit at all the second term here will be 0 and $E[F_0] = 1$; a large F_0 will be evidence of lack of fit. To make conclusions we'll compute a p-value and compare it to some level α.

To illustrate this with the automotive data I'm going to create a SLR with just disp as a predictor, this waY I can guarantee that there are multiple observations at some of the x values. To get a measure of noise we'll fit a "full" model where disp is treated as a factor – this estimates a separate coefficient for each unique value of disp. We'll then perform an ANOVA between this and the reduced model (the SLR).

```
full_mdl <- lm(mpg ~ factor(disp), data = auto_mpg)
reduced_mdl <- lm(mpg ~ disp, data = auto_mpg)
anova(full_mdl, reduced_mdl)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ factor(disp)
```

```
## Model 2: mpg ~ disp
##   Res.Df    RSS  Df Sum of Sq      F    Pr(>F)
## 1    311 4398.3
## 2    390 8378.8 -79   -3980.5 3.5628 1.003e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So, We've obtained a very small p-value and conclude that there is a lack of fit in the model. (Note: I didn't check model assumptions here, I just wanted to show you how to perform the test. But when you do this you have to make sure that the model has met the conditions necessary for the test to be valid.)

Transforming the Data

When model assumptions are not met one of the ways we can try to correct the model is by applying transformations to the response and/or the predictors. The thought is to change the relationship between the two, thus making the linear regression model more appropriate.

Transformations on the Response

When the assumption of constant variance is violated, one of the first things that should be questioned is the scale of the response values. Sometimes changing the values of y to $y' = f(y)$ can be all that is necessary to correct the assumption violations. Choosing an appropriate function to apply is an exercise of trial and error. Table 5.1 in your text provides a list, including the most common square-root and log transformations.

Recall the residual plots from last week showed slight deviation from the assumptions. In particular, the residuals associated higher predicted values showed greater variance. Here I apply a log transformation to mpg to improve the adherence to assumptions for the model:

$$\log(y) = \beta_0 + \beta_{wt}(wt) + \beta_{hp}(hp) + \beta_{disp}(disp) + \epsilon$$

In the original scale of the response this model looks like

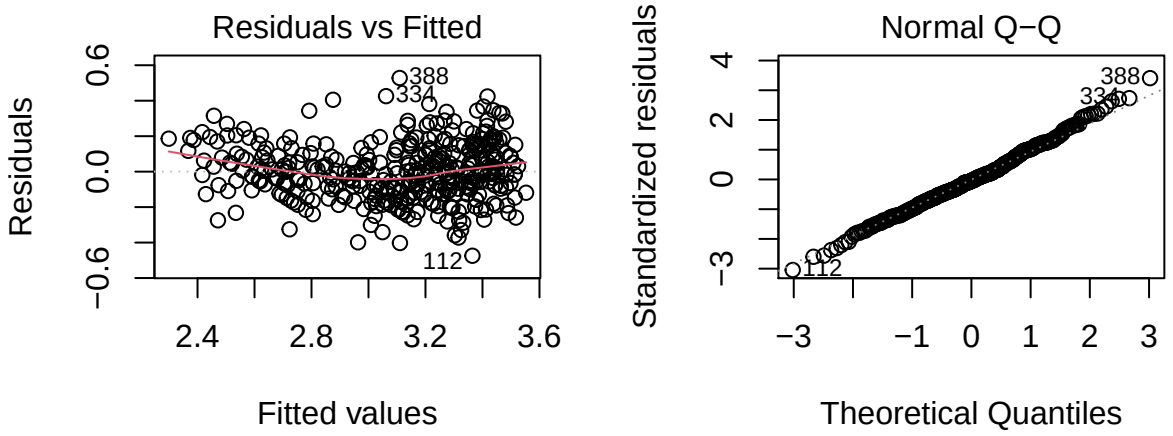
$$y = \exp\{\beta_0 + \beta_{wt}(wt) + \beta_{hp}(hp) + \beta_{disp}(disp)\} \cdot \exp\{\epsilon\}.$$

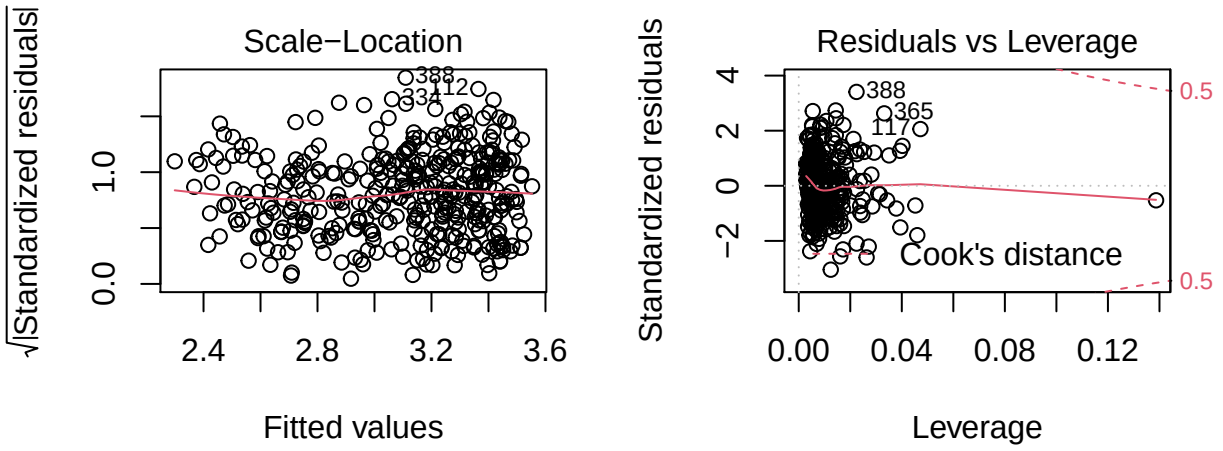
Here's the resulting model:

```
mdl_t <- lm(log(mpg) ~ wt + hp + disp, data = auto_mpg)
(mdl_t_sum <- summary(mdl_t))
```

```
##
## Call:
## lm(formula = log(mpg) ~ wt + hp + disp, data = auto_mpg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47436 -0.10001 -0.00689  0.09827  0.52821
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.063e+00  4.420e-02  91.922  < 2e-16 ***
## wt          -2.235e-04  2.633e-05  -8.487 4.54e-16 ***
## hp          -2.215e-03  4.736e-04  -4.677 4.02e-06 ***
## disp        -3.506e-04  2.433e-04  -1.441     0.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1568 on 388 degrees of freedom
## Multiple R-squared:  0.7891, Adjusted R-squared:  0.7874
## F-statistic: 483.8 on 3 and 388 DF,  p-value: < 2.2e-16
```

We see improvement with regards to R^2 and in residual analysis plots found below. In particular, the residuals vs. fitted and scale-location plots show constant variance.





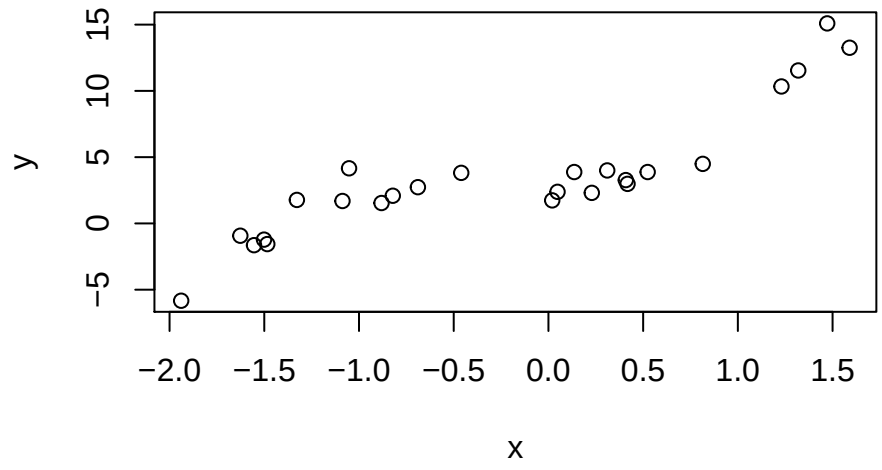
It is important to rember that predictions and their intervals should always be presented on the original scale of the response. So, for our log transformation above we would take $\hat{y} = \exp\{\hat{y'}\}$ for our predictions. Intervals need to first be computed on the transformed scale, then we exponentiate the upper and lower bounds to obtain the interval on the original scale.

Transformations on the Predictors

The choice of when to transform one of the predictors in the dataset is also done graphically; either through a scatter plot of the data, or through the partial residual plots. The idea is to transform x through $f(x)$ in the model specification. A common choice here is to replace x with a polynomial equation of some order, being careful not to overfit the data.

For a quick example, I am going to create a data set whose response is some nonlinear function of a single regressor.

```
set.seed(pi)
n = 25
x <- runif(n, -2, 2)
y <- 3 + 1.5*x^2 + 2*x^3 + rnorm(n, 0, 1)
plot(x, y)
```



Clearly the relationship between x and y here is not linear, but we can still use a linear regression model as the "linear" part of the model is with respect to the regression coefficients. So, if we specify a polynomial relationship in the model everything proceeds as normal.

Here I use the function `I()` to tell R to treat x^2 and x^3 as is and not as operators in the formula for $lm()$.

```
poly_mdl <- lm(y ~ x + I(x^2) + I(x^3))
summary(poly_mdl)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3135 -0.6604 -0.1490  0.8283  2.1181
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.8570     0.3295   8.672 2.21e-08 ***
## x             0.3619     0.5212   0.694    0.495
## I(x^2)        1.6756     0.2349   7.134 4.91e-07 ***
## I(x^3)        1.9657     0.2401   8.187 5.67e-08 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.091 on 21 degrees of freedom
## Multiple R-squared:  0.9539, Adjusted R-squared:  0.9474
## F-statistic:   145 on 3 and 21 DF,  p-value: 3.412e-14
```

Specifying the model in this way clearly leads to high correlation between predictor variables which, as we know, has the potential to lead to problems in the model. One work around is to use the `poly()` function, specifying the degree polynomial we want, then passing that object to `lm()`. Use this with caution though, as this function will return orthogonal variables resulting in different coefficients.

```
lm(y ~ poly(x, degree = 3))
```

```
##
## Call:
## lm(formula = y ~ poly(x, degree = 3))
##
## Coefficients:
##          (Intercept)  poly(x, degree = 3)1  poly(x, degree = 3)2
##                3.431                20.308                 5.046
## poly(x, degree = 3)3
##                8.932
```

There's a trade-off that you have to consider; can you accept multicollinearity or should you trade it at the loss of interpretability? The answer is context dependent.

Table 5.4 is a good reference for changing other nonlinear relationships to ones that you can build a linear regression model on.