Diagnostics and Transformations – Part 2

Contents

1 Introduction

Introduction

In this lecture, we take a closer look at transformations and regression diagnostics in the context of bivariate regression. We examine various theoretical approaches to data-driven transformation.

In the last lecture, we took a brief look at techniques for fitting a simple linear regression and examining residuals.

We saw that examining a residual plot can help us see departures from the strict linear regression model, which assumes that errors are independent, normally distributed, and show a constant conditional variance σ_ϵ^2.

Introduction

We also saw that the residual standard deviation is, in a sense, a more stable indicator of the ability of a linear model of y to predict y accurately than is the more traditional model R^2 if the model is strictly linear, because *if* the model is strictly linear over the entire range of x values, then it will be linear for any subrange of x values.

So there are some excellent reasons to want to nonlinearly transform data if there is substantial nonlinearity in the x–y scatterplot.

However, there are also reasons to linearly transform data as well.

Introduction

We shall briefly describe various approaches and rationales for linear and nonlinear transformation.

2 Linear Transformations

Linear Transformations

Linear transformations of variables do not affect the accuracy of prediction in linear regression. Any change in the x or y variables will be compensated for in corresponding changes in the β values.

However, various linear transforms can still be important for at least 3 reasons:

- Avoiding nonsensical values
- Increasing comparability
- Reducing collinearity of predictors

Avoiding Nonsensical Values by Centering

Technically, in the simple linear regression equation, the y-intercept coefficient b_0 is the value of y when $x = 0$.

In many cases, $x = 0$ does not correspond well to physical reality, as when, for example x is a person's height and y their weight. In such cases, it makes sense to at least *center* the variable x, i.e., convert it to deviation score form by subtracting its mean.

After centering the heights, a value $x = 0$ corresponds to an average height, and so the y-intercept would be interpreted as the average weight of people who are of average height.

Enhancing Comparability by Z-Score Standardization

After centering variables, they can be standardized by dividing by their standard deviation, thus converting them to z-score form.

When variables are in this form, their means are always 0 and their standard deviations are always 1, so differences are always on the same scale.

Standardizing to a Convenient Metric

Sometimes, convenience is an overriding concern, and rather then standardizing to z-score form, you choose a metric like income in tens of thousands of dollars that allows easy intrepretability.

Standardizing to a Standard Deviation of 1/2

In section 4.2, Gelman & Hill recommend standardizing numeric (not binary) "input variables" to a mean of zero and a standard deviation of 1/2, by centering, then dividing by 2 standard deviations.

They state that doing this "maintains coherence when considering binary input variables." (Binary variables coded 0,1 have a standard deviation of 1/2 when $p = 0.5$. Changing from 0 to 1 implies a shift of 2 standard deviations, which in turn results in the β weight being reduced by a factor of 2.)

Gelman & Hill explain the rationale very briefly, and this rationale is conveyed in more clarity and detail in Gelman's (2008) article in *Statistics in Medicine.*

Standardizing to a Standard Deviation of 1/2

Recall that a β conveys how much y should will increase on average for a unit increase in x. If numeric input variables are standardized, a unit increase in x corresponds to a standard deviation. However, if binary variables are coded 0,1 a unit increase from 0 to 1 corresponds to two standard deviations. Gelman & Hill see this as cause for concern, as linear regression compensates for this by decreasing the β weights on binary variables. By decreasing the standard deviation of numeric input variables to 1/2, they seek to eliminate what they see as an inherent interpretational disadvantage for binary variables.

Standardizing to a Standard Deviation of 1/2

Comment. I don't (yet!) see this argument as particularly convincing, because the standard deviation itself has meaning only in connection with a meaningful scale, which binary variables don't have.

Ultimately, a conviction to understand what numerical values actually mean in any data analytic context should trump attempts to automatize this process.

Standardizing to Eliminate Problems in Interpreting Interactions

When one of the variables is binary, and interaction effects exist, centering can substantially aid the interpretation of coefficients, because such interpretations rely on a meaningful zero point.

For example, if the model with coefficients is

$$y = 14 + 34x_1 + 12x_2 + 14x_1x_2$$

the coefficient 34 is the amount of gain in y for unit change in x *only when* $x_2 = 0$.

3 Polynomial Regression

Gelman & Hill distinguish in their terminology between *input variables* and *predictors.* A quick example:

Example 1 (Input Variables vs. Predictors). Consider a single y predicted from a single x. We might write

$$y = \beta_1 x + \beta_0 + \epsilon$$

Alternatively, we might write

$$y = \beta_1 x + \beta_2 x^2 + \beta_0 + \epsilon$$

In each case, the input variable is x. In the first case, there is also only one predictor, x, while in the second case, there are two predictors, x and x^2. In one sense, the second regression is nonlinear, because y is predicted from a nonlinear function of the input variable x. In another sense, it is linear, because y is predicted from a linear function of the predictors x and x^2.

In polynomial regression, we fit a polynomial in x of degree q to a criterion variable y.

Example 2 (Degree 3 Polynomial). For example, a polynomial regression of degree 3 would fit the model

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \epsilon$$

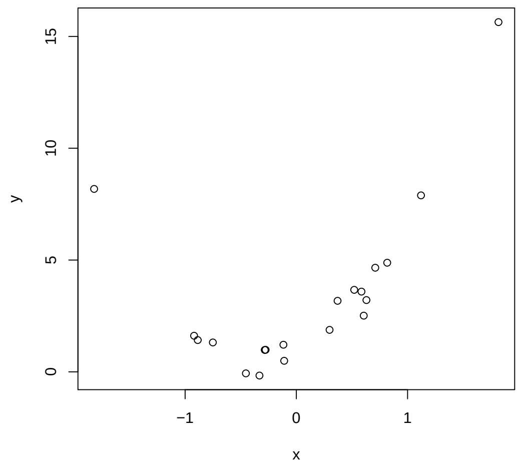
Practical Limitations

- If we use polynomial regression, we need to limit the order of the polynomial
- Perfect fit to N data points can always be achieved with a degree $N - 1$ polynomial
- Fit always improves as we add more terms
- Terms are not uncorrelated, which adds to interpretation problems, although *centering* generally will reduce the correlation between polynomial terms

Polynomial Regression

Brett Larget at Wisconsin has a nice writeup on polynomial regression. Following his example, let's create some artificial polynomial data.

```
> set.seed(12345)
> x <- rnorm(20, mean = 0, sd = 1)
> y <- 1 + 2 * x + 3 * x^2 + rnorm(20, sd = 0.5)
> plot(x, y)
```



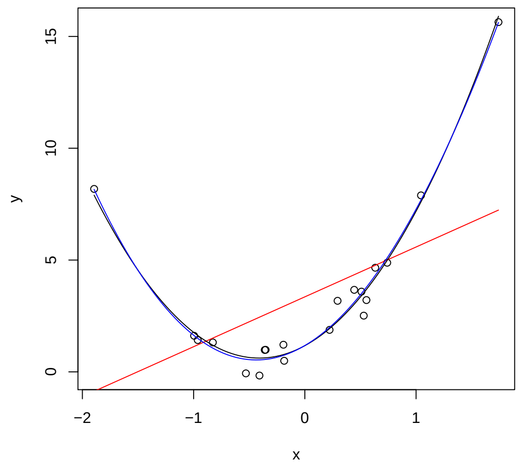
Polynomial Regression

Let's fit a sequence of polynomial models to these data. Note the use of the `I` operator. In R's model interpretation language, `x*x` has a special meaning, so you need to use this operator to convey to R that you intend an expression to be interpreted as a numerical transformation.

```
> x ← x - mean(x)
> fit0 ← lm(y ~ 1)
> fit1 ← lm(y ~ x)
> fit2 ← lm(y ~ x + I(x^2))
> fit3 ← lm(y ~ x + I(x^2) + I(x^3))
> fit4 ← lm(y ~ x + I(x^2) + I(x^3) + I(x^4))
> fit5 ← lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5))
```

Let's examine the fit of these regression lines *graphically*:

```
> ## start by plotting the data
> plot(x,y)
> ## add the line of linear fit in red
> curve(cbind(1,x) %*% coef(fit1),col='red',add=TRUE)
> ## and quadratic fit in black
> curve(cbind(1,x,x^2) %*% coef(fit2),add=TRUE)
> ## and cubic fit in blue
> curve(cbind(1,x,x^2,x^3) %*% coef(fit3),col='blue',add=TRUE)
```



It seems the cubic term adds little of consequence to the quality of fit. Check the fit with the `anova` function:

```
> anova(fit0,fit1,fit2,fit3,fit4,fit5)
```

```
Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ x
Model 3: y ~ x + I(x^2)
Model 4: y ~ x + I(x^2) + I(x^3)
Model 5: y ~ x + I(x^2) + I(x^3) + I(x^4)
Model 6: y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5)
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)    
1     19 260.978                                    
2     18 195.319  1    65.659 207.0416 8.812e-10 ***
3     17   4.760  1   190.559 600.8825 6.712e-13 ***
4     16   4.464  1     0.295   0.9316    0.3508    
5     15   4.440  1     0.024   0.0772    0.7853    
6     14   4.440  1 9.414e-05   0.0003    0.9865    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4 Orthogonal Polynomials

In situations where x is a set of equally spaced ordered categories, you can fit *orthogonal polynomials*, a set of fixed values. You can read about this in detail in CCAW. This technique breaks the linear, quadratic, cubic, etc. sources of variation into orthogonal components.

5 Empirically Motivated Nonlinear Transformations

5.1 Mosteller-Tukey Bulging Rule

Often, a nonlinear transformation of x and/or y will work wonders in improving the fit of a linear model. Mosteller and Tukey, in their classic 1977 book *Data Analysis and Regression*, list two primary advantages in linearizing a plot.

- Interpolation and interpretation are relatively easier
- Interpreting departures from good fit is easier

A Caution

Mosteller and Tukey quickly add a serious caution: When a nonlinear transformation is performed strictly on an empirical basis, extrapolation beyond the range of the data is extremely dangerous. On the other hand, if guided by strong theory, such extrapolation might be reasonable.

The Ladder of Re-Expression

Mosteller and Tukey speak of a "ladder of re-expressions" involving transformations of the form

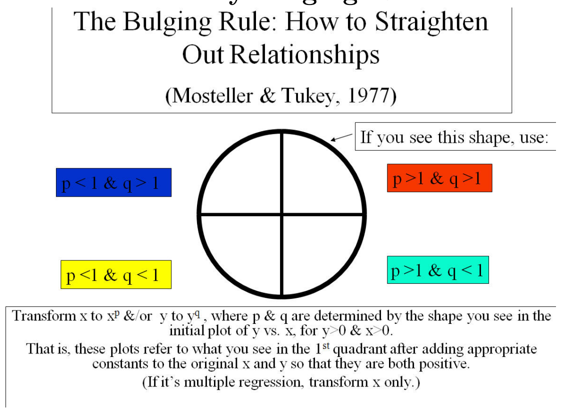
$$z = t^p \tag{1}$$

for positive values of t, and for powers p on the ladder of values $-3, -2, -1, -1/2, \#, 1/2, 1, 2, 3$, with # referring to a log transformation. On page 84, they present a graphical representation of their "bulging rule."

The Ladder of Re-Expression

Bruce Cooil of Vanderbilt's Owen School has a nice Powerpoint slide translation of the figure given by Mosteller and Tukey (1977, p. 84).

The Mosteller-Tukey Bulging Rule



The Bulging Rule: How to Straighten Out Relationships

(Mosteller & Tukey, 1977)

Transform x to x^p &/or y to y^q, where p & q are determined by the shape you see in the initial plot of y vs. x, for $y>0$ & $x>0$.

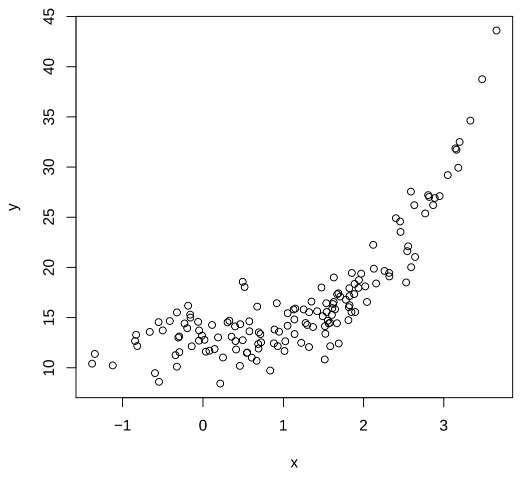
That is, these plots refer to what you see in the 1st quadrant after adding appropriate constants to the original x and y so that they are both positive.

(If it's multiple regression, transform x only.)

Using the Bulging Rule

Let's create and plot some more artificial data:

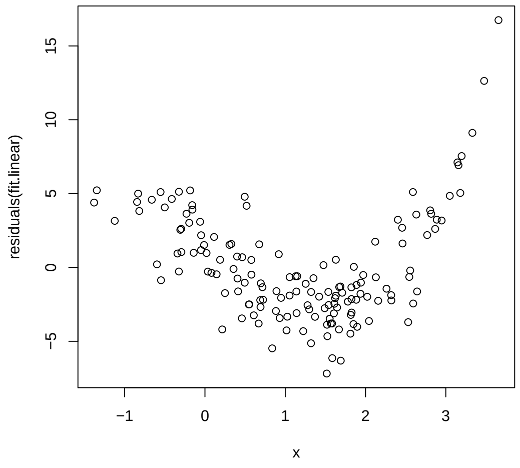
```
> set.seed(12345)
> x ← rnorm(150,1,1)
> e ← rnorm(150,0,2)
> y ← .6 *x^3 + 13 + e
> fit.linear ← lm(y~x)
> plot(x,y)
```



Using the Bulging Rule

Here is the residual plot. Although it is hard to gauge, it does not appear that residual variance changes overwhelmingly from left to right, and so transformation of x alone may be sufficient.

```
> plot(x,residuals(fit.linear))
```



Using the Bulging Rule

As you can see, the graph bulges in the direction that indicates x should move *up* the re-expression ladder. First, let's re-express x so that it it is strictly positive. The minimum value of x is

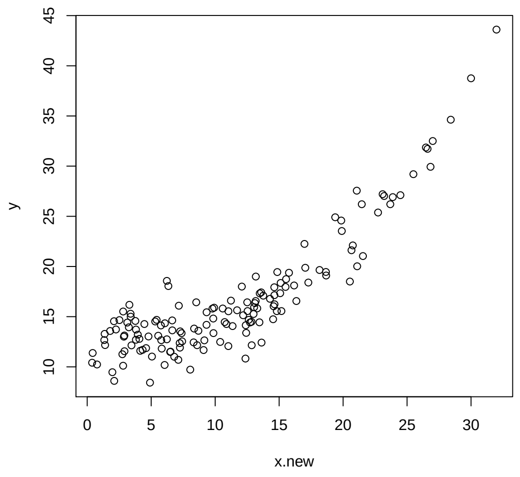
```
> min(x)
```

```
[1] -1.380358
```

Using the Bulging Rule

Hence $x + 2$ will be strictly positive. Plotting y against $(x + 2)^2$, we still see a slight rightward bulge.

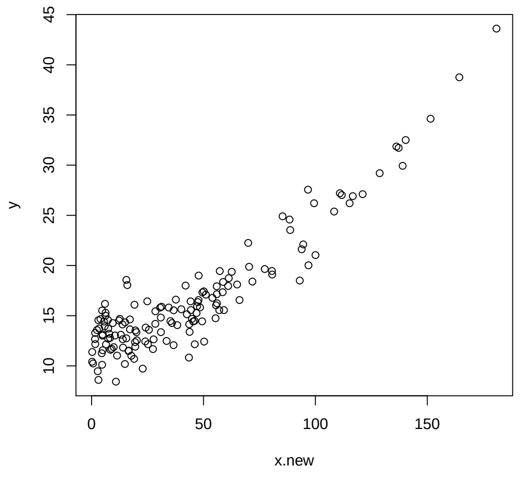
```
> x.new ← (x + 2)^2
> plot(x.new,y)
> fit.2 ← lm(y~x.new)
```



Using the Bulging Rule

Moving further up the ladder, we try plotting y against $(x+2)^3$. This looks very close to perfectly linear.

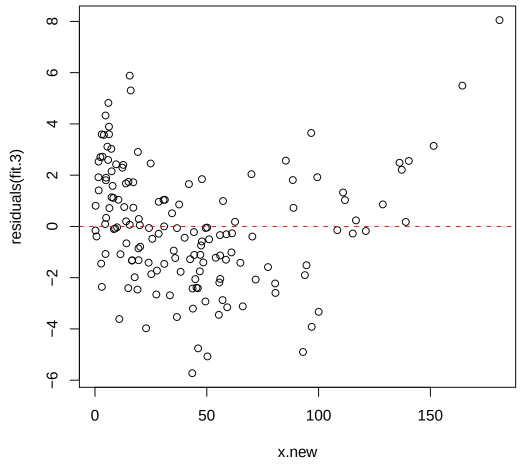
```
> x.new ← (x + 2)^3
> plot(x.new,y)
> fit.3 ← lm(y~x.new)
```



Using the Bulging Rule

Residuals look good as well.

```
> plot(x.new,residuals(fit.3))
> abline(0,0,lty=2,col='red')
```



Examining the fit statistics, we first look at the power of 2 fit:

```
> summary(fit.2)


Call:
lm(formula = y ~ x.new)

Residuals:
    Min      1Q  Median      3Q     Max
-6.5140 -1.9943 -0.4320  1.9019 12.3219

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.54601    0.44176   19.34   <2e-16 ***
x.new        0.71091    0.03315   21.45   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.914 on 148 degrees of freedom
Multiple R-squared: 0.7566,       Adjusted R-squared: 0.7549
F-statistic:   460 on 1 and 148 DF,  p-value: < 2.2e-16
```

Next, we look at the power of 3 fit. It is substantially better.

```
> summary(fit.3)


Call:
lm(formula = y ~ x.new)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-5.7268 -1.4583 -0.1245  1.6715  8.0502 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 10.535446   0.285576   36.89   <2e-16 ***
x.new        0.138311   0.004901   28.22   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 2.338 on 148 degrees of freedom
Multiple R-squared: 0.8433,	Adjusted R-squared: 0.8422 
F-statistic: 796.3 on 1 and 148 DF,  p-value: < 2.2e-16
```

5.2 Box-Cox Transformations

Box and Cox (1964) present a class of transformations of y designed to stabilize residual variances and linearize fit. Such a transformation of y can work well when the raw data show heterogeneity of error variance.

The Box-Cox family is defined in terms of a parameter λ.

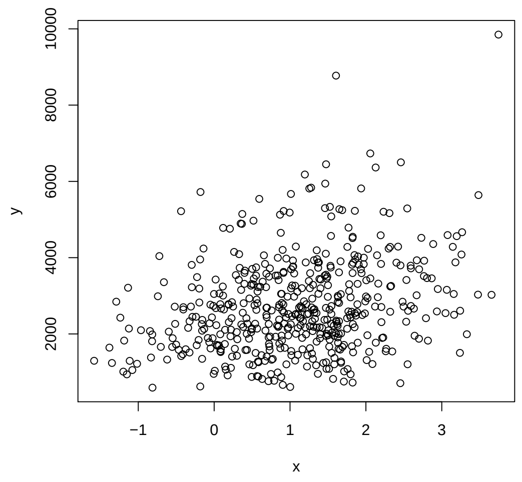
$$y_{(\lambda)} = \begin{cases} (y^\lambda - 1)/\lambda & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases} \tag{2}$$

Non-positive values of y should be avoided. Often y will be displaced by an amount α to avoid this problem.

The Box-Cox Transformation Family

Let's create and plot some slightly different artificial data:

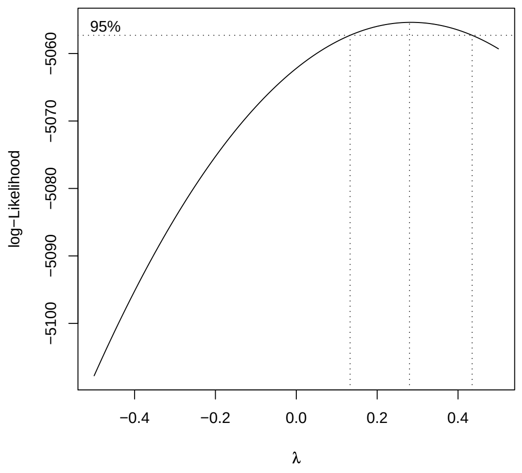
```
> set.seed(12345)
> x ← rnorm(500,1,1)
> e ← rnorm(500,0,2)
> y ← (.6 *x + 13 + e)^3
> fit.linear ← lm(y~x)
> plot(x,y)
```



Fitting a Box-Cox Transformation

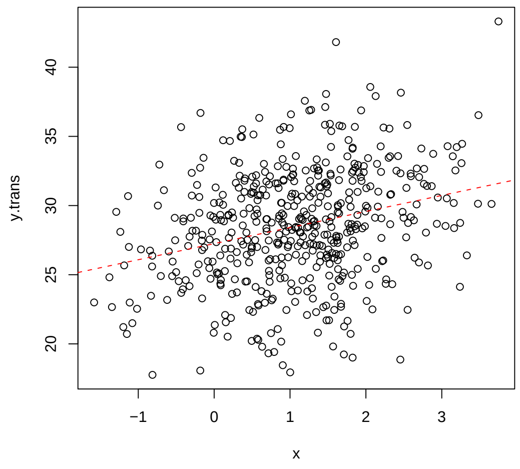
There are automated procedures available in the R library `MASS` for selecting a Box-Cox transformation. We proceed by first choosing a "reasonable" value of α (in this case, $\alpha = 0$) and finding the value of λ that maximizes the profile likelihood function. The plot shows the maximum of the function, and also indicates a 95% confidence interval for λ. Notice that the confidence interval ranges from about .15 to .42, but the profile likelihood is maximized with an approximate value of $\lambda = 0.28$.

```
> boxcox(y~x,lambda=seq(-.5,.5,.01))
```



The plot looks like this:

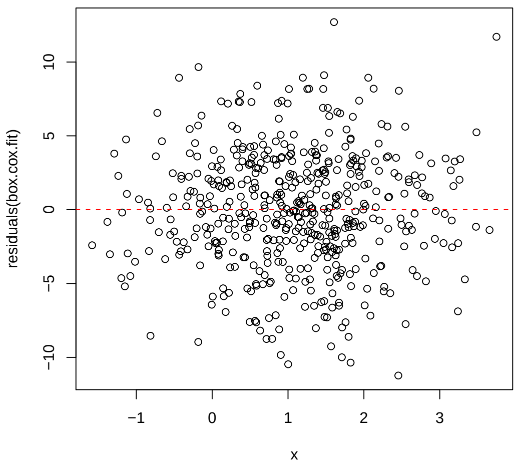
```
> trans ← function(y,lambda){(y^lambda - 1)/lambda}
> y.trans ← trans(y,.28)
> box.cox.fit ← lm(y.trans ~ x)
> plot(x,y.trans)
> abline(box.cox.fit,lty=2,col='red')
```



Fitting a Box-Cox Transformation

Here is the residual plot.

```
> plot(x,residuals(box.cox.fit))
> abline(0,0,lty=2,col='red')
```


residuals(box.cox.fit)
10
5
0
−5
−10
−1
0
1
2
3
x