Transformations and scale changes

Based on a book by Julian J. Faraway

University of Iceland

Data

We will continue to use the `savings` dataset.

```
library(faraway) # you need to install the package first
data(savings)
```

The dataframe contains the following columns:

`sr`	savings rate - personal saving divided by disposable income
`pop15`	percent population under age of 15
`pop75`	percent population over age of 75
`dpi`	per-capita disposable income in dollars
`ddpi`	percent growth rate of dpi

Where are we...

Transformation

- Transformations of the response and predictors can improve the fit and correct violations of model assumptions such as constant error variance.
- We may also consider adding additional predictors that are functions of the existing predictors like quadratic or crossproduct terms.

Transforming the response

When you use a log transformation on the response, the regression coefficients have a particular interpretation:

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

$$\hat{y} = e^{\hat{\beta}_0} e^{\hat{\beta}_1 x_1} \cdots e^{\hat{\beta}_p x_p}$$

An increase of one in x_1 would multiply the predicted response (in the original scale) by $e^{\hat{\beta}_1}$.

Thus when a log scale is used the regression coefficients can be interpreted in a multiplicative rather than the usual additive manner.

Transforming the response

- Although you may transform the response, you will probably need to express predictions in the original scale.
- This is simply a matter of back-transforming.
- In the logged model above the predictions would be $e^{\hat{y}}$.
- If the prediction confidence interval in the the logged scale was $[l, u]$ then you would use $[e^l, e^u]$. This interval will not be symmetric but this may be desirable.

Transforming the response

- Regression coefficients will need to be interpreted with respect to the transformed scale.
- There is no straightforward way of backtransforming them to values that can be interpreted in the original scale.
- You cannot directly compare regression coefficients for models where the response transformation is different.
- Difficulties of this type may dissuade one from transforming the response.

Box-Cox transformation

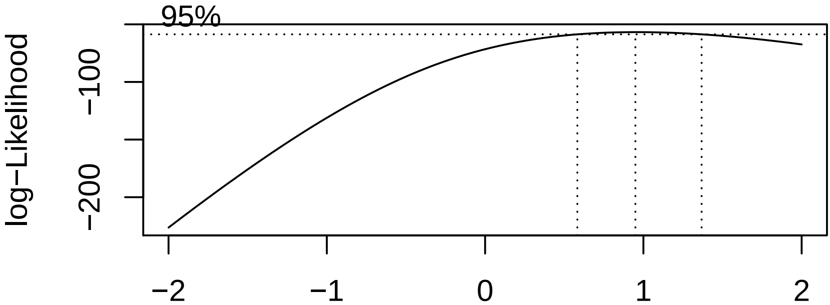
- The Box-Cox method is a popular way to determine a tranformation on the response.
- It is designed for strictly positive responses and chooses the transformation to find the best fit to the data.
- The method transforms the response $y \to t_\lambda(y)$ where the family of transformations indexed by λ is

$$t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases}$$

- λ is estimated using maximum likelihood.

Finding the value of λ using boxcox()

```
library(MASS) # you need to install the package first
g <- lm(sr~pop15+pop75+dpi+ddpi,savings)
boxcox(g,plotit=T)
```



Box-Cox transformation

- The Box-Cox method gets upset by outliers - if you find $\hat{\lambda} = 5$ then this is probably the reason - there can be little justification for actually making such an extreme transformation.
- What if some $y_i < 0$? Sometimes adding a constant to all y can work provided that the constant is small.
- If $\max_i y_i / \min_i y_i$ is small then the Box-Cox won't do anything because power transforms are well approximated by linear transformations over short intervals.
- Should the estimation of λ count as an extra parameter to be taken account of in the degrees of freedom? This is a difficult question since λ is not a linear parameter and its estimation is not part of the least squares fit.

Transforming the predictors

- You can take a Box-Cox style approach for each of the predictors, choosing the transformation to minimize the RSS.
- This takes time and furthermore the correct transformation for each predictor may depend on getting the others right too.

Transforming the predictors

Another way of generalizing the $X\beta$ part of the model is to add polynomial terms. In the one-predictor case, we have

$$y = \beta_0 + \beta_1 x + ... + \beta_d x^d + \varepsilon$$

which allows for a more flexible relationship although we usually don't believe it exactly represents any underlying reality.

Transforming the predictors

There are two ways to choose d:

- Keep adding terms until the added term is not statistically significant.
- Start with a large d — eliminate not statistically significant terms starting with the highest order term.

Warning: Do not eliminate lower order terms from the model even if they are not statistically significant.

Regression splines

- Polynomials have the advantage of smoothness but the disadvantage that each data point affects the fit globally.
- With splines we get smoothness and local influence.
- A spline is a numeric function that is piecewise-defined by polynomial functions, and which possesses a high degree of smoothness at the places where the polynomial pieces connect (which are known as knots)

"Modern" methods

- Generalized additive models (GAM)
- ACE (Alternating Conditional Expectations), AVAS (Additivity and variance stabilization), MARS (Multivariate adaptive regression splines)
- Regression trees

Transforming the predictors in R

- We can use the `poly()` function to construct orthogonal polynomials.
- We can us the `bs()` function to generate the B-spline basis matrix for a polynomial spline.

Where are we...

Changes of scale

Suppose we re-express x_i as $\frac{x_i+a}{b}$. We might want to do this because

- Predictors of similar magnitude are easier to compare. $\hat{\beta} = 3.51$ is easier to parse than $\hat{\beta} = 0.00000351$.
- A change of units might aid interpretability.
- Numerical stability is enhanced when all the predictors are on a similar scale.

Changes of scale

- Rescaling x_i with some constant $1/b$ leaves the t- and F tests and $\hat{\sigma}^2$ and R^2 unchanged but the estimate of its parameter is multiplied with b.
- Rescaling y with some constant $1/a$ leaves the t- and F tests and R^2 unchanged but all the $\hat{\beta}$-s and $\hat{\sigma}^2$ are divided by a.

Standardizing the variables

- One rather thorough approach to scaling is to convert all the variables to standard units - mean 0 and variance 1.
- This can be done using the `scale()` command.
- Such scaling has the advantage of putting all the predictors and the response on a comparable scale, which makes comparisons simpler.
- It also avoids some numerical problems that can arise when variables are of very different scales.
- The downside of this scaling is that the regression coefficients now represent the effect of a one standard unit increase in the predictor on the response in standard units — this might not always be easy to interpret.

Standardizing the variables

```
# variables on original scale
fit.1<-lm(sr~pop15+pop75+dpi+ddpi,data=savings)
summary(fit.1)


##
## Call:
## lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2422 -2.6857 -0.2488  2.4280  9.7509
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.5660865  7.3545161   3.884 0.000334 ***
## pop15       -0.4611931  0.1446422  -3.189 0.002603 **
## pop75       -1.6914977  1.0835989  -1.561 0.125530
## dpi         -0.0003369  0.0009311  -0.362 0.719173
## ddpi         0.4096949  0.1961971   2.088 0.042471 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.803 on 45 degrees of freedom
## Multiple R-squared:  0.3385, Adjusted R-squared:  0.2797
## F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904
```

Standardizing the variables

```
# x - variables scaled
fit.2<-lm(sr~I(scale(pop15))+I(scale(pop75))+I(scale(dpi))+I(scale(ddpi)),data=savings)
summary(fit.2)
```

```
##
## Call:
## lm(formula = sr ~ I(scale(pop15)) + I(scale(pop75)) + I(scale(dpi)) +
##     I(scale(ddpi)), data = savings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2422 -2.6857 -0.2488  2.4280  9.7509
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       9.6710     0.5378  17.983   <2e-16 ***
## I(scale(pop15))  -4.2207     1.3237  -3.189   0.0026 **
## I(scale(pop75))  -2.1833     1.3987  -1.561   0.1255
## I(scale(dpi))    -0.3338     0.9226  -0.362   0.7192
## I(scale(ddpi))    1.1758     0.5631   2.088   0.0425 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.803 on 45 degrees of freedom
## Multiple R-squared:  0.3385, Adjusted R-squared:  0.2797
## F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904
```

Standardizing the variables

```r
# y - variable scaled
fit.3<-lm(scale(sr)~pop15+pop75+dpi+ddpi,data=savings)
summary(fit.3)
```

```
##
## Call:
## lm(formula = scale(sr) ~ pop15 + pop75 + dpi + ddpi, data = savings)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83962 -0.59944 -0.05553  0.54191  2.17635
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.217e+00  1.641e+00   2.569   0.0136 *
## pop15       -1.029e-01  3.228e-02  -3.189   0.0026 **
## pop75       -3.775e-01  2.419e-01  -1.561   0.1255
## dpi         -7.519e-05  2.078e-04  -0.362   0.7192
## ddpi         9.144e-02  4.379e-02   2.088   0.0425 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8487 on 45 degrees of freedom
## Multiple R-squared:  0.3385, Adjusted R-squared:  0.2797
## F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904
```

Where are we...

Collinearity

- If $X^T X$ is singular, that is some predictors are linear combinations of others, we have exact collinearity and there is no unique LS estimate of $\boldsymbol{\beta}$.
- If $X^T X$ is close to singular we have approximate collinearity or multicollinearity.
- This causes serious problems with the estimation of $\boldsymbol{\beta}$ and associate quantities as well as interpretation.
- We can detect possible problems by looking at a correlation matrix of the predictors.

Collinearity

Collinearity can lead to:

- Imprecise estimates of $\boldsymbol{\beta}$.
- t-tests which fail to reveal signifficant factors.
- missing importance of predictors.

Collinearity

```r
str(longley)
```

```
## 'data.frame': 16 obs. of  7 variables:
##  $ GNP.deflator: num  83 88.5 88.2 89.5 96.2 ...
##  $ GNP         : num  234 259 258 285 329 ...
##  $ Unemployed  : num  236 232 368 335 210 ...
##  $ Armed.Forces: num  159 146 162 165 310 ...
##  $ Population  : num  108 109 110 111 112 ...
##  $ Year        : int  1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 ...
##  $ Employed    : num  60.3 61.1 60.2 61.2 63.2 ...
```

Collinearity

```
fit<-lm(Employed ~ ., data=longley)
summary(fit)
```

```
##
## Call:
## lm(formula = Employed ~ ., data = longley)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41011 -0.15767 -0.02816  0.10155  0.45539
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 **
## GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141
## GNP          -3.582e-02  3.349e-02  -1.070 0.312681
## Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 **
## Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
## Population   -5.110e-02  2.261e-01  -0.226 0.826212
## Year          1.829e+00  4.555e-01   4.016 0.003037 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3049 on 9 degrees of freedom
## Multiple R-squared:  0.9955, Adjusted R-squared:  0.9925
## F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

Collinearity

```r
cor(longley)
```

```
##              GNP.deflator       GNP Unemployed Armed.Forces Population
## GNP.deflator    1.0000000 0.9915892  0.6206334    0.4647442  0.9791634
## GNP             0.9915892 1.0000000  0.6042609    0.4464368  0.9910901
## Unemployed      0.6206334 0.6042609  1.0000000   -0.1774206  0.6865515
## Armed.Forces    0.4647442 0.4464368 -0.1774206    1.0000000  0.3644163
## Population      0.9791634 0.9910901  0.6865515    0.3644163  1.0000000
## Year            0.9911492 0.9952735  0.6682566    0.4172451  0.9939528
## Employed        0.9708985 0.9835516  0.5024981    0.4573074  0.9603906
##                   Year  Employed
## GNP.deflator 0.9911492 0.9708985
## GNP          0.9952735 0.9835516
## Unemployed   0.6682566 0.5024981
## Armed.Forces 0.4172451 0.4573074
## Population   0.9939528 0.9603906
## Year         1.0000000 0.9713295
## Employed     0.9713295 1.0000000
```

Collinearity

```
fit<-lm(Employed ~ GNP + Unemployed + Armed.Forces, data=longley)
summary(fit)
```

```
##
## Call:
## lm(formula = Employed ~ GNP + Unemployed + Armed.Forces, data = longley)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83085 -0.22306  0.01735  0.10699  1.08090
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  53.306461   0.716342  74.415  < 2e-16 ***
## GNP           0.040788   0.002207  18.485 3.49e-10 ***
## Unemployed   -0.007968   0.002134  -3.734  0.00285 **
## Armed.Forces -0.004828   0.002552  -1.892  0.08286 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4793 on 12 degrees of freedom
## Multiple R-squared:  0.9851, Adjusted R-squared:  0.9814
## F-statistic: 264.4 on 3 and 12 DF,  p-value: 3.189e-11
```