15.7: Regarding Regression Coefficients

Before moving on to discuss the assumptions underlying linear regression and what you can do to check if they're being met, there's two more topics I want to briefly discuss, both of which relate to the regression coefficients. The first thing to talk about is calculating confidence intervals for the coefficients; after that, I'll discuss the somewhat murky question of how to determine which of predictor is most important.

15.7.1 Confidence intervals for the coefficients

Like any population parameter, the regression coefficients b cannot be estimated with complete precision from a sample of data; that's part of why we need hypothesis tests. Given this, it's quite useful to be able to report confidence intervals that capture our uncertainty about the true value of b. This is especially useful when the research question focuses heavily on an attempt to find out *how* strongly variable X is related to variable Y, since in those situations the interest is primarily in the regression weight b. Fortunately, confidence intervals for the regression weights can be constructed in the usual fashion,

$$CI(b) = \hat{b} \pm (t_{crit} \; x \; SE(\hat{b}))$$

where SE($\hat{b}$) is the standard error of the regression coefficient, and t_{crit} is the relevant critical value of the appropriate t distribution. For instance, if it's a 95% confidence interval that we want, then the critical value is the 97.5th quantile of a t distribution with N−K−1 degrees of freedom. In other words, this is basically the same approach to calculating confidence intervals that we've used throughout. To do this in R we can use the `confint()` function. There arguments to this function are

- `object` . The regression model (`lm` object) for which confidence intervals are required.
- `parm` . A vector indicating which coefficients we should calculate intervals for. This can be either a vector of numbers or (more usefully) a character vector containing variable names. By default, all coefficients are included, so usually you don't bother specifying this argument.
- `level` . A number indicating the confidence level that should be used. As is usually the case, the default value is 0.95, so you wouldn't usually need to specify this argument.

So, suppose I want 99% confidence intervals for the coefficients in the `regression.2` model. I could do this using the following command:

```
confint( object = regression.2,
         level = .99)
```

```
##                   0.5 %      99.5 %
## (Intercept) 117.9755724 133.9555593
## dan.sleep   -10.4044419  -7.4960575
## baby.sleep   -0.7016868   0.7227357
```

Simple enough.

15.7.2 Calculating standardised regression coefficients

One more thing that you might want to do is to calculate "standardised" regression coefficients, often denoted β. The rationale behind standardised coefficients goes like this. In a lot of situations, your variables are on fundamentally different scales. Suppose, for example, my regression model aims to predict people's IQ scores, using their educational attainment (number of years of education) and their income as predictors. Obviously, educational attainment and income are not on the same scales: the number of years of schooling can only vary by 10s of years, whereas income would vary by 10,000s of dollars (or more). The units of measurement have a big influence on the regression coefficients: the b coefficients only make sense when interpreted in light of the units, both of the predictor variables and the outcome variable. This makes it very difficult to compare the coefficients of different predictors. Yet there are situations where you really do want to make comparisons between different coefficients. Specifically, you might want some kind of standard measure of which predictors have the strongest relationship to the outcome. This is what ***standardised coefficients*** aim to do.

The basic idea is quite simple: the standardised coefficients are the coefficients that you would have obtained if you'd converted all the variables to z-scores before running the regression.[219] The idea here is that, by converting all the predictors to z-scores, they all go into the regression on the same scale, thereby removing the problem of having variables on different scales. Regardless of what the original variables were, a β value of 1 means that an increase in the predictor of 1 standard deviation will produce a corresponding 1 standard deviation increase in the outcome variable. Therefore, if variable A has a larger absolute value of β than variable B, it is deemed to have a stronger relationship with the outcome. Or at least that's the idea: it's worth being a little cautious here, since this does rely very heavily on the assumption that "a 1 standard deviation change" is fundamentally the same kind of thing for all variables. It's not always obvious that this is true.

Leaving aside the interpretation issues, let's look at how it's calculated. What you could do is standardise all the variables yourself and then run a regression, but there's a much simpler way to do it. As it turns out, the β coefficient for a predictor X and outcome Y has a very simple formula, namely

$$\beta_X = b_X \times \frac{\sigma_X}{\sigma_Y}$$

where σ_X is the standard deviation of the predictor, and σY is the standard deviation of the outcome variable Y. This makes matters a lot simpler. To make things even simpler, the `lsr` package includes a function `standardCoefs()` that computes the β coefficients.

```
standardCoefs( regression.2 )
```

```
##                      b        beta
## dan.sleep  -8.95024973 -0.90474809
## baby.sleep  0.01052447  0.00217223
```

This clearly shows that the `dan.sleep` variable has a much stronger effect than the `baby.sleep` variable. However, this is a perfect example of a situation where it would probably make sense to use the original coefficients b rather than the standardised coefficients β. After all, my sleep and the baby's sleep are *already* on the same scale: number of hours slept. Why complicate matters by converting these to z-scores?