I have shown that we can build linear combinations of model parameters. For example, if we have two parameter estimates, say Q and W, with standard errors respectively equal to \(\sigma\_Q\) and \(\sigma\_W\), we can build a linear combination as follows:

\[Z = AQ + BW + C\]

where A, B and C are three coefficients. The standard error for this combination can be obtained as:

\[ \sigma\_Z = \sqrt{ A^2 \sigma^2\_Q + B^2 \sigma^2\_W + 2AB \sigma\_{QW} }\]

In biology, nonlinear transformations are much more frequent than linear transformations. Nonlinear transformations are, e.g., \(Z = exp(Q + W)\), or, \(Z = 1/(Q + W)\). What is the standard error for these nonlinear transformations? This is not a complex problem, but the solution may be beyond biologists with an average level of statistical proficiency. It is named the ‘delta method’ and it provides the so called ‘delta standard errors’. I thought it might be useful to talk about it, by using a very simple language and a few examples.

**Example 1: getting the half-life of a herbicide**

A herbicide has proven to follow a first order degradation kinetic in soil, with constant degradation rate \(k = -0.035\) and standard error equal to \(0.00195\). What is the half-life (\(T\_{1/2}\)) of this herbicide and its standard error?

Every pesticide chemist knows that the half-life (\(T\_{1/2}\)) is derived by the degradation rate, according to the following equation:

\[T\_{1/2} = \frac{\log(0.5)}{k}\]

Therefore, the half-life for our herbicide is:

Thalf <- log(0.5)/-0.035

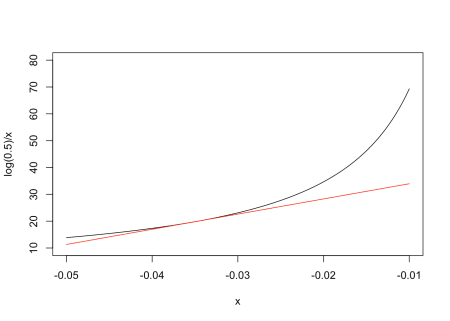
Thalf

## [1] 19.80421

But … what is the standard error of this half-life? There is some uncertainty around the estimate of \(k\) and it is clear that such an uncertainty should propagate to the estimate of \(T\_{1/2}\); unfortunately, the transformation is nonlinear and we cannot use the expression given above for linear transformations.

**The basic idea**

The basic idea behind the delta method is that most of the simple nonlinear functions, which we use in biology, can be locally approximated by the tangent line through a point of interest. For example, our nonlinear half-life function is \(Y = \log(0.5)/X\) and, obviously, we are interested in the point where \(X = k = -0.035\). In the graph below, we have represented our nonlinear function (in black) and its tangent line (in red) through the above point: we can see that the approximation is fairly good in the close vicinity of \(X = -0.035\).



What is the equation of the tangent line? In general, if the nonlinear function is \(G(X)\), you may remember from high school that the slope \(m\) of the tangent line is equal to the first derivative of \(G(X)\), that is \(G'(X)\). You may also remember that the equation of a line with slope \(m\) through the point \(P(X\_1, Y\_1)\) is \(Y – Y\_1 = m(X – X\_1)\). As \(Y\_1 = G(X\_1)\), the tangent line has equation:

\[Y = G(X\_1) + G'(X\_1)(X – X\_1)\]

**We need the derivative!**

In order to write the equation of the red line in the Figure above, we need to consider that \(X\_1 = -0.035\) and we need to be able to calculate the first derivative of our nonlinear half-life fuction. I am not able to derive the expression of the first derivative for all nonlinear functions and it was a relief for me to discover that R can handle this task in simple ways, e.g. by using the function ‘D()’. For our case, it is:

D(expression(log(0.5)/X), "X")

## -(log(0.5)/X^2)

Therefore, we can use this R function to calculate the slope \(m\) of the tangent line in the figure above:

X <- -0.035

m <- eval( D(expression(log(0.5)/X), "X") )

m

## [1] 565.8344

We already know that \(G(-0.035) = 19.80421\). Therefore, we can write the equation of the tangent line (red line in the graph above):

\[Y = 19.80421 + 565.8344 \, (X + 0.035)\]

that is:

\[Y = 19.80421 + 565.8344 \, X + 565.8344 \cdot 0.035 = 39.60841 + 565.8344 \, X\]

**Replacing a curve with a line**

Now, we have two functions:

1. the original nonlinear half-life function \(Y = \log(0.5)/X\)$
2. a new linear function (\(Y = 39.60841 + 565.8344 \, X\)), that is a very close approximation to the previous one, at least near to the point \(X = -0.035\), which we are interested in.

Therefore, we can approximate the former with the latter! If we use the linear function, we see that the half-life is:

39.60841 + 565.8344 \* -0.035

## [1] 19.80421

which is what we expected. The advantage is that we can now use the low of propagation of errors to estimate the standard error (see the first and second equation in this post):

\[ \sigma\_{ \left[ 39.60841 + 565.8344 \, X \right]} = \sqrt{ 562.8344^2 \, \sigma^2\_X}\]

Here we go:

sqrt( m^2 \* (0.00195 ^ 2) )

## [1] 1.103377

**In general…**

If we have a nonlinear transformation \(G(X)\), the standard error for this transformation is approximated by knowing the first derivative \(G'(X)\) and the standard error of \(X\):

\[\sigma\_{G(X)} \simeq \sqrt{ [G'(X)]^2 \, \sigma^2\_X }\]

**Example 2: a back-transformed count**

A paper reports that the mean number of microorganisms in a substrate, on a logarithmic scale, was \(X\_1 = 5\) with standard error \(\sigma = 0.84\). It is easy to derive that the actual number of micro-organisms was \(\exp{5} = 148.4132\); what is the standard error of the back-transformed mean?

The first derivative of our nonlinear function is:

D(expression(exp(X)), "X")

## exp(X)

and thus the slope of the tangent line is:

X <- 5

m <- eval( D(expression(exp(X)), "X") )

m

## [1] 148.4132

According to the function above, the standard error for the back-transformed mean is:

sigma <- 0.84

sqrt( m^2 \* sigma^2 )

## [1] 124.6671

**Example 3: Selenium concentration in olive drupes**

The concentration of selenium in olive drupes was found to be \(3.1 \, \mu g \,\, g^{-1}\) with standard error equal to 0.8. What is the intake of selenium when eating one drupe? Please, consider that one drupe weights, on average, 3.4 g (SE = 0.31) and that selenium concentration and drupe weight show a covariance of 0.55.

The amount of selenium is easily calculated as:

X <- 3.1; W = 3.4

X \* W

## [1] 10.54

Delta standard errors can be obtained by considering the partial derivatives for each of the two variables:

mX <- eval( D(expression(X\*W), "X") )

mW <- eval( D(expression(X\*W), "W") )

and combining them as follows:

sigmaX <- 0.8; sigmaW <- 0.31; sigmaXW <- 0.55

sqrt( (mX^2)\*sigmaX^2 + (mW^2)\*sigmaW^2 + 2\*X\*W\*sigmaXW )

## [1] 4.462726

For those of you who would like to get involved with matrix notation: we can reach the same result via matrix multiplication (see below). This might be easier when we have more than two variables to combine.

der <- matrix(c(mX, mW), 1, 2)

sigma <- matrix(c(sigmaX^2, sigmaXW, sigmaXW, sigmaW^2), 2, 2, byrow = T)

sqrt( der %\*% sigma %\*% t(der) )

## [,1]

## [1,] 4.462726

**The delta method with R**

In R there is a shortcut function to calculate delta standard errors, that is available in the ‘car’ package. In order to use it, we need to have:

1. a named vector for the variables that we have to combine
2. an expression for the transformation
3. a variance-covariance matrix

For the first example, we have:

obj <- c("k" = -0.035)

sigma <- matrix(c(0.00195^2), 1, 1)

library(car)

## Loading required package: carData

deltaMethod(object = obj, g="log(0.5)/k", vcov = sigma)

## Estimate SE 2.5 % 97.5 %

## log(0.5)/k 19.80421 1.103377 17.64163 21.96678

For the second example:

obj <- c("X1" = 5)

sigma <- matrix(c(0.84^2), 1, 1)

deltaMethod(object = obj, g="exp(X1)", vcov = sigma)

## Estimate SE 2.5 % 97.5 %

## exp(X1) 148.4132 124.6671 -95.92978 392.7561

For the third example:

obj <- c("X" = 3.1, "W" = 3.4)

sigma <- matrix(c(0.8^2, 0.55, 0.55, 0.31^2), 2, 2, byrow = T)

deltaMethod(object = obj, g="X \* W", vcov = sigma)

## Estimate SE 2.5 % 97.5 %

## X \* W 10.54 4.462726 1.793218 19.28678

The function ‘deltaMethod()’ is very handy to be used in connection with model objects, as we do not need to provide anything, but the transformation function. But this is something that requires another post!

However, two final notes relating to the delta method need to be pointed out here:

1. the delta standard error is always approximate;
2. if the original variables are gaussian, the transformed variable, usually, is not gaussian.

Thanks for reading!