Workflow Confidence Interval Estimation

Jeremiah MF Kelly

15/02/2021

The parameters are first estimated using a gradient descent method (Polyak, B.T. Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5):1–17, 1964.)

To find the confidence intervals of the parameter estimate we use the delta method (Cox, C. (2005). Delta Method. In Encyclopedia of Biostatistics (eds P. Armitage and T. Colton). https://doi.org/10.1002/0470011815.b2a15029).

This paper provides an easy to read introduction Jay M. Ver Hoef ((2012) Who Invented the Delta Method?, The American Statistician, 66:2, 124-127, DOI: 10.1080/00031305.2012.687494). The method uses a Taylor expansion of a function to approximate the variance of that function, the expansion is as follows,

$$Y = f(\boldsymbol{\theta}) \tag{1}$$

$$Y = f(\hat{\boldsymbol{\theta}}_i) + f'(\hat{\boldsymbol{\theta}}_i)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i) + \frac{1}{2}f''(\hat{\boldsymbol{\theta}}_i)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i)^2 + \ldots, \tag{2}$$

where $f'(\hat{\boldsymbol{\theta}}_i)$ and $f''(\hat{\boldsymbol{\theta}}_i)$ are the first and second derivatives of the function at $\hat{\boldsymbol{\theta}}_i$. The expansion can be as large as is wanted plus a remainder, my method assumes that R in the following is small.

$$Y = f(\hat{\boldsymbol{\theta}}_i) + f'(\hat{\boldsymbol{\theta}}_i)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i) + R, \tag{3}$$

$$Y \simeq f(\hat{\boldsymbol{\theta}}_i) + f'(\hat{\boldsymbol{\theta}}_i)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i), \tag{4}$$

$$Y - f(\hat{\boldsymbol{\theta}}_i) \simeq f'(\hat{\boldsymbol{\theta}}_i)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i), \tag{5}$$

The left handside shows the difference for each threshold between the model and the observed value for a given set of parameters $\hat{\boldsymbol{\theta}}_i$. Squaring these values, summing and dividing by the degrees of freedom gives us the mean squared error of the residuals. This implies that the mean squared error is equal to the gradient vector squared multiplied by the standard deviation of the parameter estimates.

$$(Y - f(\hat{\boldsymbol{\theta}}_i))^2 \simeq (f'(\hat{\boldsymbol{\theta}}_i)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i))^2, \tag{6}$$

$$\sum((Y - f(\hat{\boldsymbol{\theta}}_i))^2) \simeq \sum((f'(\hat{\boldsymbol{\theta}}_i)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i))^2), \tag{7}$$

$$\sum \frac{(Y - f(\hat{\boldsymbol{\theta}}_i))^2}{df} \simeq f'(\hat{\boldsymbol{\theta}}_i)^2 \sum \frac{(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_i)^2}{df}, \tag{8}$$

if we square both sides and take the expectation (i.e. divide by degrees of freedom) we establish that the variance is

$$Var(Y_{res}) = [f'(\hat{\boldsymbol{\theta}}_i)]^2 \sigma^2_{\hat{\boldsymbol{\theta}}_i}. \tag{9}$$

Application

The Mahroo Lamb Pugh model can be written,

$$Y = CT + CC * exp\left(\frac{x}{\tau}\right) + S2 * H(\alpha, x) + S3 * H(\beta, x), \tag{10}$$

where $H(., x)$ is a switch function, I use a version of the logistic function. However, McGwin uses a step function. I use a continuous function so that it is easier to differentiate.

$$H(\rho, x) = \frac{x - \rho}{1 + exp(-k(x - \rho))} \tag{11}$$

The vector representing the gradient $\boldsymbol{G} = f'(\hat{\boldsymbol{\theta}}_i)$ is used to calculate the variance-covariance matrix along with the standard error of the residuals for a given $\hat{\boldsymbol{\theta}}_i$

$$Var(Y_{res}) = \frac{\sum((Y - f(\hat{\boldsymbol{\theta}}_i))^2}{df}, \tag{12}$$

$$Var(Y_{res}) = (\boldsymbol{G}'\boldsymbol{G}) Var_{lin}(\hat{\boldsymbol{\theta}}_i) \tag{13}$$

$$Var_{lin}(\hat{\boldsymbol{\theta}}_i) = (\boldsymbol{G}'\boldsymbol{G})^{-1} Var(Y_{res}) \tag{14}$$

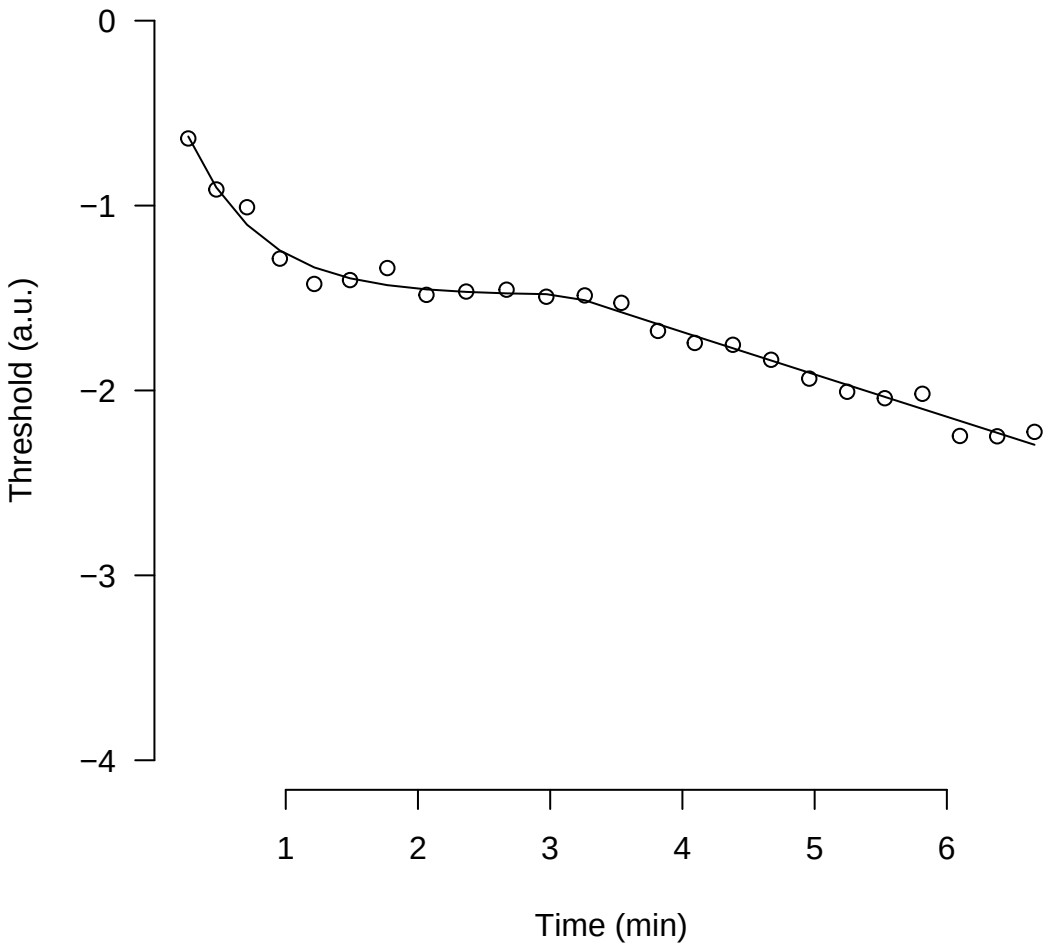
The standard errors of the parameter estimates are given by the square root of the matrix (Var_{lin}) diagonal There is a summary of the method in R here http://sia.webpopix.org/nonlinearRegression.html#standard-errors-of-the-parameter-estimates

Worked example in R

This is for the five parameter model, the seven parameter model uses the same method

```
# crosscheck SE for pk = 241
theta = c(-1.4854467,  1.3885495,  0.5477909, -0.2287327,  3.1289670 )
time <- x <- c(0.26296667, 0.4753    , 0.70723333, 0.9555    , 1.21553333,
       1.48633333, 1.7684    , 2.06418333, 2.36468333, 2.67003333,
       2.97075   , 3.26135   , 3.5383    , 3.81528333, 4.0935    ,
       4.38153333, 4.67076667, 4.95933333, 5.24596667, 5.53056667,
       5.81501667, 6.09876667, 6.38078333, 6.66281667)

thrs <- y <- c(-0.637, -0.913, -1.009, -1.287, -1.424, -1.403, -1.338, -1.483,
       -1.465, -1.455, -1.493, -1.486, -1.526, -1.678, -1.743, -1.753,
       -1.834, -1.936, -2.007, -2.042, -2.018, -2.246, -2.248, -2.224)
        P5c <- function(P,x){
          P[1] + P[2] * exp(-x/P[3]) + P[4] * (x - P[5])/(1 + exp(-2 *
+     20 * (x - P[5])))
        }
      par(las = 1, bty ='n')
       plot(time, thrs, ylim =c(-4,0), xlab ="Time (min)", ylab ="Threshold (a.u.)")
       lines(time,P5c(theta,x) )
```



the gradient can be found programmatically

```
Gfn <- deriv(y~(ct + cc * exp(-x/tau) + S2 * (x - alpha)/(1 + exp(-2 *
+     20 * (x - alpha))) ), c("ct", "cc", "tau", "S2", "alpha"), function(ct,cc,tau,S2,alpha, x){})

G <- attr(Gfn(theta[1],theta[2],theta[3],theta[4],theta[5], x),"gradient")

sigma2 = (sum((y - P5c(theta,x))**2)/(length(time)-5))
SEs <- sqrt(sigma2 * diag(solve(t(G)%*%G)))
round(SEs,4)
```

```
##     ct     cc    tau     S2  alpha
## 0.0314 0.1371 0.0839 0.0142 0.1843
```

using inbuilt functions

```
nlm1 <- nls(y~(ct + cc * exp(-x/tau) + S2 * (x - alpha)/(1 + exp(-2 *
+     20 * (x - alpha)))), start=c(ct = -1.5, cc = 1.3, tau = 0.6, S2 = -0.1, alpha = 3))
summary(nlm1)
```

```
##
## Formula: y ~ (ct + cc * exp(-x/tau) + S2 * (x - alpha)/(1 + exp(-2 * +20 *
##     (x - alpha))))
```

```
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## ct    -1.48545    0.03144 -47.248  < 2e-16 ***
## cc     1.38855    0.13715  10.124 4.31e-09 ***
## tau    0.54779    0.08391   6.528 2.98e-06 ***
## S2    -0.22873    0.01418 -16.131 1.52e-12 ***
## alpha  3.12897    0.18434  16.974 6.14e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0546 on 19 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 9.588e-07
```