Delta method
Appendix F of *WNC* (esp. F.4) & Appendix B of *CW*

Once you've produced an estimate of one or more parameters in a model of interest, you often want to use those estimates to obtain estimates of the mean and variance of transformations of those parameter estimates. Our attention here will be on estimating the variance of the transformed variable. One common approach is to use the *Delta method*, which propagates the errors or uncertainty about the estimated parameters into the variance of the transformed quantity based on those estimates or random variables. From page B-7 of *CW: "…* the Delta method rests on the assumption the first-order Taylor expansion around the parameter value is effectively linear over the range of values likely to be encountered."

Examples of Transformations of Interest

$$\widehat{Diff} = \hat{S}_A - \hat{S}_B$$

$$\widehat{MLS} \text{ (or Mean Life Span)} = -\frac{1}{\ln(\hat{S})}$$

$$\hat{\lambda}_1 = \frac{\hat{N}_2}{\hat{N}_1}$$

$$\widehat{NestSuccess} = \widehat{DSR}^{35}$$

$$\hat{S} = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 \cdot PatchSize)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 \cdot PatchSize)}$$

The Delta Method

As shown below, the Delta method uses a first-order Taylor series expansion. Appendix B of *CW* goes into great detail on the approach and I will not repeat much of that information here. Instead, the material here focuses on the concept. We are interested in bringing two types of information together:

(1) How uncertain are we of the estimates being used in the equation that generates the transformed quantity? This is measured by the appropriate values in the variance-covariance matrix.

(2) How much does that uncertainty matter to the transformed quantity? This is measured using calculus, specifically derivatives.

The two types of information are multiplied together (by matrix multiplication).

$$\text{var}(Y) = \text{var}(g(X)) = (g'(\mu))^2 \text{ var}(X)$$

The trick is getting the derivatives but we can use software to help us find them so it's not too hard.

Examples

A. $\widehat{Diff} = \hat{S}_A - \hat{S}_B$

1. Obtain the partial derivatives of transformation with respect to each of the random variables involved:

$$\frac{d\widehat{Diff}}{d\hat{S}_A} = 1 \cdot \hat{S}_A^{1-1} = 1; \quad \frac{d\widehat{Diff}}{d\hat{S}_B} = -1 \cdot \hat{S}_B^{1-1} = -1$$

2. Multiply the vector of derivatives by the variance-covariance matrix in appropriate fashion.

$$\hat{\sigma}^2_{\widehat{Diff}} = \begin{pmatrix} 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \hat{\sigma}^2[\hat{S}_A] & \hat{\sigma}[\hat{S}_A, \hat{S}_B] \\ \hat{\sigma}[\hat{S}_A, \hat{S}_B] & \hat{\sigma}^2[\hat{S}_B] \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\hat{\sigma}^2_{\widehat{Diff}} = \begin{pmatrix} 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 0.0059525 & 0.0001484673 \\ 0.0001484673 & 0.0015154 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix} = 0.007170965$$

$$\sqrt{\hat{\sigma}^2_{\widehat{Diff}}} = 0.08468155$$

R code –

```
sigma=matrix(c(0.0059525, 0.0001484673, 0.0001484673, 0.0015154),2,2)
derivs=matrix(c(1,-1),2,1)
VarDiff=t(derivs)%*%sigma%*%derivs
sP = 0.5770599
sG = 0.7699526
Diff= sG - sP
lciDiff=Diff-1.96*sqrt(VarDiff)
uciDiff=Diff+1.96*sqrt(VarDiff)
c(lciDiff, Diff, uciDiff)
```

or … with ‘msm’ package installed, which works out the derivatives for you

```
library(msm)
sigma=matrix(c(0.0059525, 0.0001484673, 0.0001484673, 0.0015154),2,2)
sP = 0.5770599
sG = 0.7699526
Diff= sG - sP
seDiff=deltamethod(~x1-x2,c(sG,sP),sigma,ses=TRUE)
lciDiff=Diff-1.96*seDiff
uciDiff=Diff+1.96*seDiff
c(lciDiff, Diff, uciDiff)                    # 0.02691686 0.19289270 0.35886854
```

B. $\hat{\lambda}_1 = \frac{\hat{N}_2}{\hat{N}_1}$

1. $\frac{d\hat{\lambda}_1}{d\hat{N}_1} = -\frac{\hat{N}_2}{\left(\hat{N}_1\right)^2}$; $\frac{d\hat{\lambda}_1}{d\hat{N}_2} = \frac{1}{\hat{N}_1}$

2. $\hat{\sigma}^2_{\hat{\lambda}_1} = \begin{pmatrix} -\frac{\hat{N}_2}{\left(\hat{N}_1\right)^2} & \frac{1}{\hat{N}_1} \end{pmatrix} \cdot \begin{pmatrix} \hat{\sigma}^2[\hat{\lambda}_1] & \hat{\sigma}[\hat{\lambda}_1, \hat{\lambda}_2] \\ \hat{\sigma}[\hat{\lambda}_1, \hat{\lambda}_2] & \hat{\sigma}^2[\hat{\lambda}_2] \end{pmatrix} \cdot \begin{pmatrix} -\frac{\hat{N}_2}{\left(\hat{N}_1\right)^2} \\ \frac{1}{\hat{N}_1} \end{pmatrix} = 0.0183023$

$\hat{\sigma}_{\hat{\lambda}_1} = \sqrt{\hat{\sigma}^2_{\hat{\lambda}_1}} = 0.1353$

R code –

```
N1=117.04
N2=67.20
L1=N2/N1
sigma=matrix(c(575.75, 144.02, 144.02, 226.29),2,2)
seLambda=deltamethod(~x2/x1,c(N1,N2),sigma,ses=TRUE)
lciL1=L1-1.96*seLambda
uciL1=L1+1.96*seLambda
c(lciL1, L1, uciL1)     # 0.3090021 0.5741627 0.8393232
```

C. $\widehat{NestSuccess} = \widehat{DSR}^{35}$

1. $\frac{d\widehat{NS}}{d\widehat{DSR}} = 35 \cdot \widehat{DSR}^{34}$

2. $\hat{\sigma}^2_{\widehat{NS}} = \left(35 \cdot \widehat{DSR}^{34}\right)^2 \cdot \left(\hat{\sigma}^2_{\widehat{DSR}}\right) = 0.000308$

$\hat{\sigma}_{\widehat{NS}} = \sqrt{\hat{\sigma}^2_{\widehat{NS}}} = 0.01755$

R code –

```
DSR=0.9528288
seDSR=0.002592525^2
seNS=deltamethod(~x1^35,DSR,seDSR,ses=TRUE)
NS=DSR^35
lciNS=NS-1.96*seNS
```

```
uciNS=NS+1.96*seNS
c(lciNS, NS, uciNS)          # 0.1498985 0.1842980 0.2186976
```

D. $\hat{S} = \dfrac{\exp(\hat{\beta}_0 + \hat{\beta}_1 \cdot PatchSize)}{1+\exp(\hat{\beta}_0 + \hat{\beta}_1 \cdot PatchSize)}$

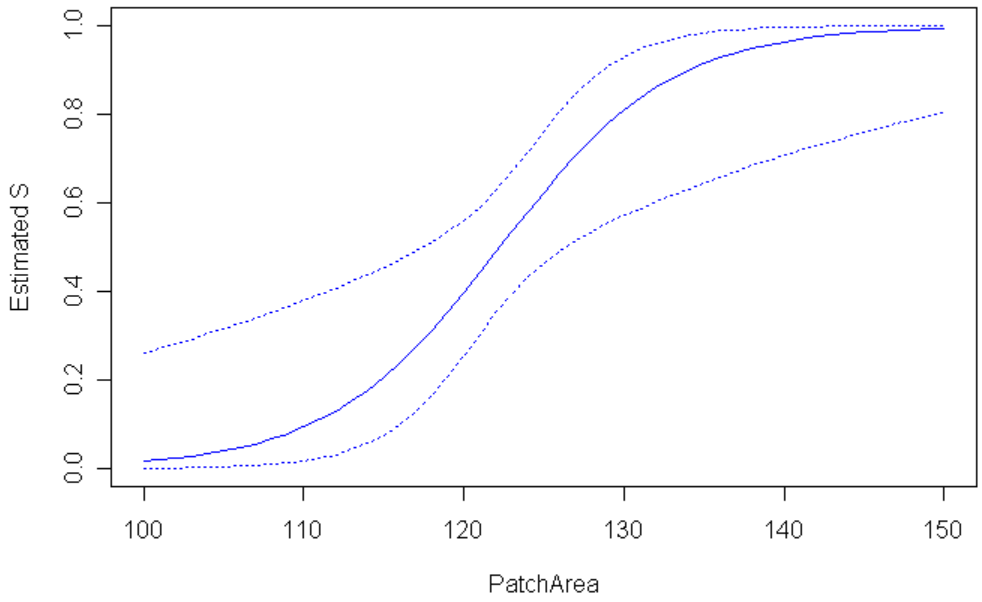
$\hat{\beta}_0 = -22.72412$ $\hat{\beta}_1 = 0.18589$

$$\hat{\Sigma} = \begin{bmatrix} 71.0620388 & -0.579228143 \\ -0.579228143 & 0.004726825 \end{bmatrix}$$

R code –

```
out2=matrix(0,51,9)
for (i in 1:51) {
  PA=99+i
  out2[i,1]=PA  # store value of Patch Area in output matrix
  out2[i,2]=betas[1]+betas[2]*PA     # store est. of ln(S/(1-S))
  formula <- sprintf("~x1+%f*x2", PA)  # build formula to include 'PA'
  # Store se(ln(S/(1-S)))
  out2[i,3]=deltamethod(as.formula(formula),betas,sigma,ses=TRUE)
  out2[i,4]=out2[i,2]-1.96*out2[i,3]   # store est of lcl for ln(S/(1-S))
  out2[i,5]=out2[i,2]+1.96*out2[i,3]   # store est of lcl for ln(S/(1-S))
  out2[i,6]=1/(1+exp(-(out2[i,2])))   # store est of S
  formula <- sprintf("~1/(1+exp(-(x1+%f*x2)))", PA)
  # Store se(S)
  out2[i,7]=deltamethod(as.formula(formula),betas,sigma,ses=TRUE)
  out2[i,8]=1/(1+exp(-(out2[i,4])))   # store lcl for est of S
  out2[i,9]=1/(1+exp(-(out2[i,5])))   # store ucl for est of S
}
colnames(out2)=c('PatchArea','logOddsS','seLogOdds','lclLO','uclLO',
   'estS','seS','lclS','uclS')
out2=as.data.frame(out2)

with(data=out2,plot(PatchArea,estS,'l',col='blue',ylim=c(0,1),ylab='Estimated S'))
points(out2$PatchArea,out2$lclS,type='l',lty=3,col='blue')
points(out2$PatchArea,out2$uclS,type='l',lty=3,col='blue')
```



Other Methods of Developing Confidence Intervals do exist and sometimes are preferred (see B-7, *CW*)