**Section 2.2.4 – Probability of success**

As shown earlier, the estimate for π is



To find a confidence interval for π, consider the logistic regression model with only one explanatory variable x:

  or 

Wald interval

To find a Wald confidence interval for π, we need to first find an interval for β0 + β1x (or equivalently for logit(π)):



where



and , , and  are obtained from the estimated covariance matrix for the parameter estimates.

You may have seen in another class that for random variables Y1 and Y2 and constants a and b, we have



To find the Wald confidence interval for π, we use the  transformation:



For a model with p explanatory variables, the interval is



where



and x0 = 1. Verify on your own that the interval given for p explanatory variables is the same as the original interval given for p = 1 explanatory variable.

Profile likelihood ratio interval

Profile LR confidence intervals for π can be found as well, but they can be much more difficult computationally to find than for OR. This is because a larger number of parameters are involved.

For example, the one explanatory variable model  is a linear combination of β0 and β1. The numerator of -2log(Λ) involves maximizing the likelihood function with a constraint for this linear combination.

Like for the Wald interval, an interval for  is found first and then the  transformation is applied to its lower and upper limits.

The mcprofile package (not in the default installation of R) provides a general way to compute profile likelihood ratio intervals. Because computations are more difficult, one should be on the lookout for potential problems. These problems may come about through warning messages printed. However, most warning messages will not be of concern if taken in the appropriate context. One approach to the computational difficulty is to compute both Wald and profile LR intervals. If the intervals are substantially different, then the profile LR computations may we questionable.

Example: Placekicking (Placekick.R, Placekick.csv)

Consider the model with only distance as the explanatory variable:



where the results from glm() are saved in the object mod.fit.

To estimate the probability of success for a distance of 20 yards, I used the following code before:

> linear.pred <- mod.fit$coefficients[1] +

 mod.fit$coefficients[2]\*20

> linear.pred

(Intercept)

 3.511547

> exp(linear.pred)/(1+exp(linear.pred))

(Intercept)

 0.9710145

> as.numeric(exp(linear.pred)/(1+exp(linear.pred)))

 #Removes label

[1] 0.9710145

There are easier ways to perform this calculation. First, the 3rd observation in the data is for a distance of 20 yards. This leads to the third value in mod.fit$fitted.values to be the estimated probability of success at this distance:

> head(placekick$distance == 20)

[1] FALSE FALSE TRUE FALSE TRUE FALSE

> mod.fit$fitted.values[3] #3rd obs. distance = 20

3 0.9710145

The best way to perform the calculation is to use the predict() function:

> predict.data <- data.frame(distance = 20)

> predict(object = mod.fit, newdata = predict.data, type =

 "link")

1 3.511547

> predict(object = mod.fit, newdata = predict.data, type =

 "response")

1 0.9710145

The predict() function is a generic function, so predict.glm() is actually used to perform the calculations. Also, notice the argument value of type = "link" calculates  (equivalently, ).

Note that one could create the data frame for newdata inside of predict():

predict(object = mod.fit, newdata = data.frame(distance = 20), type = "response")

Wald interval

To find the Wald confidence interval, we can calculate components of the interval for  through adding arguments to the predict() function:

> alpha <- 0.05

> linear.pred <- predict(object = mod.fit, newdata =

 predict.data, type = "link", se = TRUE)

> linear.pred

$fit

1. 3.511547

$se.fit

[1] 0.1732707

$residual.scale

[1] 1

> pi.hat <- exp(linear.pred$fit) / (1 + exp(linear.pred$fit))

> CI.lin.pred <- linear.pred$fit + qnorm(p = c(alpha/2, 1-

 alpha/2))\*linear.pred$se

> CI.pi <- exp(CI.lin.pred)/(1+exp(CI.lin.pred))

> CI.pi

[1] 0.9597647 0.9791871

> round(data.frame(predict.data, pi.hat, lower = CI.pi[1],

 upper = CI.pi[2], digits = 4)

 distance pi.hat lower upper

1 20 0.971 0.9598 0.9792

The 95% Wald confidence interval for π is 0.9598 < π < 0.9792; thus, the probability of success for the placekick is quite high at a distance of 20 yards.

A different Wald confidence interval is



where  is found through a delta method approximation (see Appendix B). The predict() function can be used to calculate  by using the type = "response" argument value along with se = TRUE.

> pi.hat <- predict(object = mod.fit, newdata = predict.data, type = "response", se = TRUE)

> pi.hat

$fit

 1

0.9710145

$se.fit

 1

0.00487676

$residual.scale

[1] 1

> ci.pi2 <- pi.hat$fit + qnorm(p = c(alpha/2, 1-

 alpha/2))\*pi.hat$se

> data.frame(predict.data, pi.hat = round(pi.hat$fit,4),

 lower = round(ci.pi2[1],4), upper = round(ci.pi2[2],4))

 distance pi.hat lower upper

1 20 0.971 0.9615 0.9806

This interval is quite close to what we had before. In fact, this interval will be close in general as long as there is a large sample size and/or π is not close to 0 or 1. HOWEVER, I recommend not using this interval because its limits can be greater than 1 or less than 0.

Using the original Wald confidence interval equation again, we can also calculate more than one interval at a time and include more than one explanatory variable. Below is an example using the estimated model



that we found earlier and then saved the results from glm() in an object called mod.fit2:

> predict.data <- data.frame(distance = c(20,30), change =

 c(1, 1))

> predict.data

 distance change

1 20 1

2 30 1

> alpha <- 0.05

> linear.pred <- predict(object = mod.fit2, newdata =

 predict.data, type = "link", se = TRUE)

> CI.lin.pred.x20 <- linear.pred$fit[1] + qnorm(p =

 c(alpha/2, 1-alpha/2)) \* linear.pred$se[1]

> CI.lin.pred.x30 <- linear.pred$fit[2] + qnorm(p =

 c(alpha/2, 1-alpha/2)) \* linear.pred$se[2]

> round(exp(CI.lin.pred.x20)/(1+exp(CI.lin.pred.x20)), 4)

 #CI for distance = 20

[1] 0.9404 0.9738

> round(exp(CI.lin.pred.x30)/(1+exp(CI.lin.pred.x30)), 4)

 #CI for distance = 30

[1] 0.8493 0.9159

Profile likelihood ratio interval

Below is the code for the profile LR interval for a distance of 20 yards:

> library(package = mcprofile)

> K <- matrix(data = c(1, 20), nrow = 1, ncol = 2)

> K

 [,1] [,2]

[1,] 1 20

> #Calculate -2log(Lambda)

> linear.combo <- mcprofile(object = mod.fit, CM = K)

> #CI for beta\_0 + beta\_1 \* x

> ci.logit.profile <- confint(object = linear.combo, level = 0.95)

> ci.logit.profile

 mcprofile - Confidence Intervals

level: 0.95

adjustment: single-step

 Estimate lower upper

C1 3.51 3.19 3.87

> names(ci.logit.profile)

[1] "estimate" "confint" "CM" "quant"

[5] "alternative" "level" "adjust"

> exp(ci.logit.profile$confint)/(1 +

 exp(ci.logit.profile$confint))

 lower upper

C1 0.9603165 0.979504

Of course, you will need to have installed the mcprofile package first before using this code! Below is an explanation of the code:

1. The K object is a 1×2 matrix containing the coefficients on the β’s in . Essentially, we will be performing matrix multiplication of the form



1. The mcprofile() function uses this matrix in its CM argument (CM stands for “contrast matrix”). This function calculates -2log(Λ) for a large number of possible values of the linear combination.
2. The confint() function uses all of these values to find the 95% confidence interval for β0 + β120.
3. Finally, I use the  transformation to find the confidence interval for π.

The 95% interval for π is 0.9603 < π < 0.9795, which is similar to the Wald interval due to the large sample size.

Additional examples of working with the mcprofile package are given in the corresponding program. Specifically, I show how to

* Calculate an interval for π at more than one distance
* Control the familywise confidence level at a specified level
* Construct an interval for π using a model that contains both distance and change
* Use the wald() function to find the same Wald intervals as found earlier (even if problems occur with running mcprofile(), you should be able to use the wald() function)

emmeans package

The user contributed package emmeans provides another way to compute these Wald intervals in this situation. This is a package that can be used in a wide variety of situations outside of categorical data analysis. In particular, it is used frequently for multiple comparisons when using ANOVA methods.

Example: Placekicking (Placekick.R, Placekick.csv)

For the model that includes only the distance of the placekick, we can estimate π and compute confidence intervals for it using the following code:

> library(package = emmeans)

> predict.data2 <- list(distance = 20)

> emmeans(object = mod.fit, specs = ~ distance, at = predict.data2, type = "response", level = 0.95)

distance prob SE df asymp.LCL asymp.UCL

 20 0.971 0.00488 Inf 0.96 0.979

Comments:

* This is a little easier than using the predict() function. However, as we examine this package later, we will see that there are quirks with its use that lessen its appeal at times.
* The level argument specifies the confidence level.
* The object and type arguments represent the same arguments as for predict().
* The specs argument specifies the variable(s) at those values we wish to estimate probabilities. The format is similar to the right side of a formula argument, with a + separating different variables when needed, and does not have to include all variables in the model.
* The optional at argument lists the values of these variables in specs at which we wish to do the computations, similar to newdata in predict(), but its value should be structured in a list format. For example, rather than using

newdata = data.frame(distance = c(20,30), change = c(1, 1))

as with predict() for the model that contains before distance and change, emmeans() uses

at = list(distance = c(20,30), change = 1)

where all possible combinations of the explanatory variable values are created automatically. When a model contains more variables than are given in specs, results are averaged over levels of the omitted variables.

Overall, the emmeans package does not provide much of a benefit for its use here. However, introducing it in this simple setting will help us later on for more complicated settings when it does provide a benefit.