**Chapter 2 – Analyzing a binary response – Part 2**

Section 1.1 discusses how to estimate and make inferences about a single probability of success π. Section 1.2 generalizes this discussion to the situation of two probabilities of success that are now dependent on a level of a group. Chapter 2 now completes the generalization to a situation where there are many different possible probabilities of success to estimate and perform inferences upon. Furthermore, this chapter allows us to quantify how an explanatory variable with many possible levels (perhaps continuous rather than categorical) affects the probability of success. These generalizations are made through the use of binary regression models.

**Section 2.1 – Linear models**

Review of normal linear regression models

Yi = β0 + β1xi1 + β2xi2 + … + βpxip + εi

where εi ~ independent N(0, σ2) and i = 1, …, n

Note that

E(Yi) = β0 + β1xi1 + β2xi2 + … + βpxip

E(Yi) is what one would expect Yi  to be on average for a set of xi1, xi2, …, xip values. Also, note that this model implies that Yi ~ independent N(E(Yi), σ2).

If βk (one of the β’s above) is equal to 0, this says there is no linear relationship between the corresponding explanatory variable xk and the response variable. If βk > 0, there is a positive relationship, and if βk < 0, there is a negative relationship. All of these statements are with respect to the other explanatory variables in the model remaining constant.

Regression models for a binary response

Let Yi be a binary response variable for i = 1, …, n, where a 1 denotes a success and a 0 denotes a failure.

Suppose Yi has a Bernoulli distribution again as in Chapter 1, but now with the probability of success parameter πi. Thus, Yi ~ independent Bernoulli(πi). Notice that the probability of success can be different for i = 1, …, n observations. Potentially, there could be n different parameters then we need to estimate!

We can simplify the number of parameters that we need to estimate by using a linear model of the form

 E(Yi) = β0 + β1xi1

where I am using just one explanatory variable to simplify the explanation. Because E(Yi) is πi, we could also write the model as

πi = β0 + β1xi1

Therefore, instead of potentially having n different parameters to estimate, we now only have two!!!

To estimate the parameters, we should not proceed as we would with normal linear regression models for the following reasons:

* Yi is binary here, but Yi had a continuous distribution in normal linear regression.
* Var(Yi) = πi(1 – πi) for a Bernoulli random variable; thus, the variance potentially changes for each Yi. With normal linear regression, Var(Yi) = Var(εi) = σ2 is constant for i = 1, …, n.

We estimate the β’s through using maximum likelihood estimation. The likelihood function is



where πi = β0 + β1xi1. Maximizing the likelihood function leads to the maximum likelihood estimates of β0 and β1.

Unfortunately, there is still a problem – πi = β0 + β1xi1 is not constrained to be within 0 and 1. For particular values of β0, β1, and xi1, πi may end be greater than 1 or less than 0.

**Section 2.2 – Logistic regression models**

There are a number of solutions to prevent πi from being outside the range of a probability. Most solutions rely on non-linear transformations to prevent these types of problems from occurring. The most commonly used transformation results in the logistic regression model:



Notice that  so that the numerator is always less than the denominator. Thus, 0 < πi <1.

The logistic regression model can also be written as



through using some algebra. Notice that the left-hand side is the log transformation of the odds of a success! This will be very important for us later when interpreting the effect an explanatory variable has on the response variable.

Comments:

* + - The  transformation is often referred to as the logit transformation. Thus, the most compact way that people write the model as is



* + - The  part of the model is often referred to as the linear predictor.
		- We can write the model without the i subscript when we want to state the model in general:

 and 

Obviously, this leads to some notational ambiguity with what we had in Section 1.1 for π, but the meaning should be obvious within the context of the problem.

Example: Plot of π vs. x (PiPlot.R)

When there is only one explanatory variable, β0 = 1, and β1 = 0.5 (or -0.5), a plot of π vs. x looks like the following:



We can make the following generalizations:

* 0 < π < 1
* When β1 > 0, there is a positive relationship between x1 and π. When β1 < 0, there is a negative relationship between x1 and π.
* The shape of the curve is somewhat similar to the letter s.
* Above π = 0.5 is a mirror image of below π = 0.5.
* The slope of the curve is dependent on the value of x1. We can show this mathematically by taking the derivative with respect to x1: 

R code:

> par(mfrow = c(1,2))

> beta0<-1

> beta1<-0.5

> curve(expr = exp(beta0+beta1\*x)/(1+exp(beta0+beta1\*x)),

 xlim = c(-15, 15), col = "black", main = expression(pi

 == frac(e^{1+0.5\*x[1]}, 1+e^{1+0.5\*x[1]})), xlab =

 expression(x[1]), ylab = expression(pi))

> beta0<-1

> beta1<--0.5

> curve(expr = exp(beta0+beta1\*x)/(1+exp(beta0+beta1\*x)),

 xlim = c(-15, 15), col = "black", main = expression(pi

 == frac(e^{1-0.5\*x[1]}, 1+e^{1-0.5\*x[1]})), xlab =

 expression(x[1]), ylab = expression(pi))

Questions:

* What happens to the β1 = 0.5 plot when β1 is increased?
* What happens to the β1 = 0.5 plot when β1 is decreased to be close to 0?
* Suppose a plot of logit(π) vs. x1 was made. What would the plot look like?

**Section 2.2.1 – Parameter estimation**

Maximum likelihood estimation is used to estimate the parameters of the model. As shown earlier, the likelihood function is



but now



in the above expression. Similar to Section 1.1, we can find the log likelihood function:



Taking derivatives with respect to β0, …, βp, setting them equal to 0, and then solving for the parameters lead to the MLEs. These parameter estimates are denoted by  …, . Corresponding estimates of π are



Unfortunately, there are no closed form expressions that can be written out for  …,  except in very simple cases. The MLEs instead are found through using iterative numerical procedures. Appendix B of the book gives a simple example of the Newton-Raphson procedure, one of these iterative numerical procedures, for finding the MLE of π in a homogeneous population setting (i.e., Section 1.1). We will use a procedure called iteratively reweighted least squares (IRLS) to find the maximum likelihood estimates.

Without going into all of the details behind IRLS, initial estimates for the parameters, say  …, , are found. Weighted least squares estimation (see Chapter 11 of my STAT 870 notes; weights are based on ) is used to find a “better” set of parameter estimates.

If the new parameter estimates, say  …, , are very close to  …, , the iterative numerical procedure is said to “converge” and  …,  are used as the MLEs  …, . If the new parameter estimates  …,  are not very close to  …, , weighted least squares estimation is used again with new weights. This iterative process continues until convergence or a prior-specified maximum number of iterations is reached.

The glm() function computes the parameter estimates.

Question: If the prior-specified maximum number of iterations limit is reached, should the last set of parameter estimates be used as  …, ?

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

This example is motivated by the work that I did for my MS report and Bilder and Loughin (*Chance*, 1998).

The purpose of this and future examples involving this data is to estimate the probability of success for a placekick. Below are the explanatory variables to be considered:

* Week: week of the season
* Distance: Distance of the placekick in yards
* Change: Binary variable denoting lead-change (1) vs. non-lead-change (0) placekicks; successful lead-change placekicks are those that change which team is winning the game.
* Elap30: Number of minutes remaining before the end of the half with overtime placekicks receiving a value of 0
* PAT: Binary variable denoting the type of placekick where a point after touchdown (PAT) is a 1 and a field goal is a 0
* Type: Binary variable denoting dome (0) vs. outdoor (1) placekicks
* Field: Binary variable denoting grass (1) vs. artificial turf (0) placekicks
* Wind: Binary variable for placekicks attempted in windy conditions (1) vs. non-windy conditions (0); I define windy as a wind stronger than 15 miles per hour at kickoff in an outdoor stadium

The response variable is referred to as “Good” in the data set. It is a 1 for successful placekicks and a 0 for failed placekicks.

There are 1,425 placekick observations from the 1995 NFL season that are within this data set. Below is how the data is read into R:

> placekick<-read.table(file = "C:\\data\\placekick.csv",

 header = TRUE, sep = ",") #read.csv() would work too

> head(placekick)

 week distance change elap30 PAT type field wind good

1 1 21 1 24.7167 0 1 1 0 1

2 1 21 0 15.8500 0 1 1 0 1

3 1 20 0 0.4500 1 1 1 0 1

4 1 28 0 13.5500 0 1 1 0 1

5 1 20 0 21.8667 1 0 0 0 1

6 1 25 0 17.6833 0 0 0 0 1

For this particular example, we are only going to use the distance explanatory variable to estimate the probability of a successful placekick. Thus, our logistic regression model is



where Y is the good response variable and x1 denotes the distance in yards for the placekick. Less formally, we will also write the model as



Below is how we use R to estimate the model with the glm() function:

> mod.fit<-glm(formula = good ~ distance, family =

 binomial(link = logit), data = placekick)

> mod.fit

Call: glm(formula = good ~ distance, family = binomial(link = logit), data = placekick)

Coefficients:

(Intercept) distance

 5.8121 -0.1150

Degrees of Freedom:1424 Total (i.e. Null); 1423 Residual

Null Deviance: 1013

Residual Deviance:775.7 AIC: 779.7

The estimated logistic regression model is



Note that the function gets its name from “generalized linear model”. This is a general class of linear models which includes logistic regression models. At the end of this chapter, I will formally define this general class.

Question: What happens to the estimated probability of success as the distance increases?

There is actually much more information stored within the mod.fit object than showed so far. Through the use of the names() function, we obtain the following list of items:

> names(mod.fit)

 [1] "coefficients" "residuals"

 "fitted.values" "effects" "R"

 [6] "rank" "qr" "family"

 "linear.predictors" "deviance"

[11] "aic" "null.deviance" "iter"

 "weights" "prior.weights"

[16] "df.residual" "df.null" "y"

 "converged" "boundary"

[21] "model" "call" "formula"

 "terms" "data"

[26] "offset" "control" "method"

 "contrasts" "xlevels"

> mod.fit$coefficients

(Intercept) distance

 5.8120798 -0.1150267

To see a summary of all the information in mod.fit, we can use the summary() function:

> summary(object = mod.fit)

Call: glm(formula = good ~ distance, family = binomial(link = logit), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

 -2.7441 0.2425 0.2425 0.3801 1.6092

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 5.812080 0.326277 17.81 <2e-16 \*\*\*

distance -0.115027 0.008339 -13.79 <2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.43 on 1424 degrees of freedom

Residual deviance: 775.75 on 1423 degrees of freedom

AIC: 779.75

Number of Fisher Scoring iterations: 5

Now is a good time for a reminder of why R is often referred to as an “object oriented language”. As discussed in Appendix A (introduction to R), every object in R has a class associated with it. The classes for mod.fit are:

> class(mod.fit)

[1] "glm" "lm"

Associated with each class, there are a number of “method” functions:

> methods(class = glm)

 [1] add1.glm\* anova.glm confint.glm\*

 cooks.distance.glm\* deviance.glm\*

 [6] drop1.glm\* effects.glm\*

 extractAIC.glm\* family.glm\* formula.glm\*

[11] influence.glm\* logLik.glm\*

 model.frame.glm nobs.glm\* predict.glm

[16] print.glm residuals.glm rstandard.glm

 rstudent.glm summary.glm

[21] vcov.glm\* weights.glm\*

 Non-visible functions are asterisked

Notice all of the method functions have the class name at their end. For example, there is a summary.glm() function. When the generic function summary() is run, R first finds the class of the object and then checks to see if there is a summary.glm() function. Because the function exists, this method function completes the main calculations.

The purpose of generic functions is to use a familiar language set with any object. For example, we frequently want to summarize data or a model (summary()), compute confidence intervals (confint()), and find predictions (predict()), so it is convenient to use the same language set no matter the application.

We can find the estimated probability of success for a particular distance using:



For example, the probability of success at a distance of 20 is 0.97:

> 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

The estimated probability of success for a distance of 50 yards is 0.52 (code not in program):

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

 mod.fit$coefficients[2]\*50

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

[1] 0.515182

Using this method to estimate the probability of success, we can now plot the model with the curve() function:

> curve(expr = exp(mod.fit$coefficients[1] +

 mod.fit$coefficients[2]\*x) / (1 +

 exp(mod.fit$coefficients[1] +

 mod.fit$coefficients[2]\*x)), col = "red", xlim = c(18,

 66), ylab = expression(hat(pi)), xlab = "Distance",

 main = "Estimated probability of success for a

 placekick", panel.first = grid())



Note that the exact code given above is not in the corresponding program to this example. Later, we will learn about a little easier way to use the curve() function to construct this plot.

If more than one explanatory variable is included in the model, the variable names can be separated by “+” symbols in the formula argument. For example, suppose we include the change variable in addition to distance in the model:

> mod.fit2<-glm(formula = good ~ change + distance, family

 = binomial(link = logit), data = placekick)

> summary(mod.fit2)

Call: glm(formula = good ~ change + distance, family = binomial(link = logit), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

 -2.7061 0.2282 0.2282 0.3750 1.5649

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 5.893181 0.333184 17.687 <2e-16 \*\*\*

change -0.447783 0.193673 -2.312 0.0208 \*

distance -0.112889 0.008444 -13.370 <2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.4 on 1424 degrees of freedom

Residual deviance: 770.5 on 1422 degrees of freedom

AIC: 776.5

Number of Fisher Scoring iterations: 6

> mod.fit2$coefficients

(Intercept) change distance

 5.8931801 -0.4477830 -0.1128887

The estimated logistic regression model is



While we use the glm() function to find our MLEs for convenience, one could also find these estimates by programming in the log likelihood function and maximizing it ourselves. This is especially useful to know how to do for other “non-standard” problems that may occur in problems faced outside of this class. Please see my book for an example of how to find the MLEs for the  model without using glm(). The code uses a VERY nice function called optim(), which performs a number of different iterative numerical procedures. I have found this function to be very useful in my research!

Variance-Covariance matrix

The estimated variance-covariance matrix for  …,  has the usual form:



Thus, the (1,1) element is the estimated variance of , the (1,2) element is the estimate covariance of  and  … . I will more simply call this matrix a “covariance matrix” for the remainder of the notes.

This matrix is found through using the same likelihood based methods discussed in Chapter 1.

If  is the MLE for θ, we can say that



for a large sample, where



Now, we need to account for not only 1 MLE, but for p + 1 MLEs. The result is a matrix of second partial derivatives (a Hessian matrix) of the log-likelihood function evaluated at the parameter estimates, and multiplying this resulting matrix by -1. For example, if there are only parameters β0 and β1, the matrix is



In the end, the matrix in general ends up having the nice form of  where



and  is a n×n matrix with  on the diagonal and 0’s elsewhere.

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

The vcov() function produces the estimated covariance matrix:

> vcov(mod.fit)

 (Intercept) distance

(Intercept) 0.106378825 -2.604526e-03

distance -0.002604526 6.950142e-05

> vcov(mod.fit)[2,2] #Var^(beta^\_1)

[1] 6.950142e-05

From the summary(mod.fit) output earlier, we have

> summary(object = mod.fit)

Call: glm(formula = good ~ distance, family = binomial(link = logit), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

 -2.7441 0.2425 0.2425 0.3801 1.6092

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 5.812079 0.326158 17.82 <2e-16 \*\*\*

distance -0.115027 0.008337 -13.80 <2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.43 on 1424 degrees of freedom

Residual deviance: 775.75 on 1423 degrees of freedom

AIC: 779.75

Number of Fisher Scoring iterations: 5

If we square the items in the “Std. Error” column of the output, we obtain the corresponding variances:

> 0.326158^2

[1] 0.106379

> 0.008337^2

[1] 6.950557e-05

> summary(mod.fit)$coefficients[,2]^2

 (Intercept) distance

1.064568e-01 6.953996e-05

If we wanted to go through the actual matrix calculations ourselves (this would not typically be of interest!), we obtain the same matrix:

> pi.hat<-mod.fit$fitted.values

> V<-diag(pi.hat\*(1-pi.hat))

> X<-cbind(1, placekick$distance)

> solve(t(X)%\*%V%\*%X)

 [,1] [,2]

[1,] 0.106456753 -2.606250e-03

[2,] -0.002606250 6.953996e-05

Binomial data with more than 1 trial

So far, our data has consisted of Y1, …, Yn responses from Bernoulli distributions with parameters πi for i = 1, …, n. Alternatively, we may actual observe W1, …, WJ from binomial distributions with nj trials and probability of success πj where j = 1, …, J. Note that I am using j’s here to help differentiate this notation from the Bernoulli observation notation.

The likelihood function is



The log likelihood function is



This is the same likelihood function as if we used a Bernoulli form of the data (think of each wj as being the sum of particular Yi’s which have the exact same explanatory variable values) EXCEPT for . Because this additional term is constant for all possible values of our parameters (there is no πj) in it, we will still obtain the same parameter estimates as if we worked with a Bernoulli form of the data.

Note that the maximized value of the log likelihood function will be different due to . Also, we will see in Chapter 5 that there will be differences when evaluating how well the model fits the data.

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

To illustrate that the Bernoulli and binomial forms of a data set result in the same parameter estimates, I am going to first transform the placekicking data set to a binomial form. For this example, I will only use distance as the explanatory variable.

The aggregate() function is used to find the number of successes (wj) and number of observations for each distance (nj):

> w<-aggregate(formula = good ~ distance, data = placekick,

 FUN = sum)

> n<-aggregate(formula = good ~ distance, data = placekick,

 FUN = length)

> w.n<-data.frame(distance = w$distance, success = w$good,

 trials = n$good, proportion = round(w$good/n$good,4))

> head(w.n)

 distance success trials proportion

1 18 2 3 0.6667

2 19 7 7 1.0000

3 20 776 789 0.9835

4 21 19 20 0.9500

5 22 12 14 0.8571

6 23 26 27 0.9630

For example, there are 2 successes out of 3 trials at a distance of 18 yards, which results in an observed proportion of successes of 2/3 ≈ 0.6667. Note that the reason for the large number of observations at 20 yards is because most PATs are attempted from this distance.

Below is the code used to estimate the model:

> mod.fit.bin<-glm(formula = success/trials ~ distance,

 weight = trials, family = binomial(link = logit), data

 = w.n)

> summary(mod.fit.bin)

Call: glm(formula = success/trials ~ distance, family = binomial(link = logit), data = w.n, weights = trials)

Deviance Residuals:

 Min 1Q Median 3Q Max

-2.0373 -0.6449 -0.1424 0.5004 2.2758

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 5.812080 0.326277 17.81 <2e-16 \*\*\*

distance -0.115027 0.008339 -13.79 <2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 282.181 on 42 degrees of freedom

Residual deviance: 44.499 on 41 degrees of freedom

AIC: 148.46

Number of Fisher Scoring iterations: 4

The estimated model is the same as before:



Section 1.2 data and logistic regression models

The 2×2 contingency tables given in Section 1.2 can be rewritten in a data frame format so that a logistic regression model can be fit to them. Please see the homework for an example.

**Section 2.2.2 – Hypothesis tests for regression parameters**

We often want to assess the importance of an explanatory variable or groups of explanatory variables. One way to make this assessment is through using hypothesis tests. For example, suppose we are interested in the rth explanatory variable xr in the model



If βr = 0, we see that xr would be excluded from the model. Thus, we are interested in hypothesis tests of the form:

H0: βr = 0

Ha: βr ≠ 0

Alternatively, we could state the hypotheses as:



Notice that the null hypothesis model terms are all included within the alternative hypothesis model. In other words, the null hypothesis model is a special case of the alternative hypothesis model. For this reason, the null hypothesis model is often referred to as a reduced model and the alternative hypothesis model is often referred to as a full model. The purpose of this section is to examine two ways that hypothesis tests of this form can be performed.

Wald test

The Wald statistic is



to test H0: βr = 0 vs. Ha: βr ≠ 0. For a large sample, the test statistic has an approximate standard normal distribution if the null hypothesis of βr = 0 is true. Thus, reject the null hypothesis if we observe a test statistic value that is “unusual” for a standard normal distribution. We define unusual to be . The p-value is  where Z ~ N(0,1). Wald test statistics and p-values are automatically provided for individual β parameters using code like summary(mod.fit).

The Wald test can also be performed for more than one parameter at the same time. However, because the Wald test has similar problems to those we saw in Chapter 1 for Wald tests and confidence intervals, we are not going to pursue how to perform these types of tests.

Likelihood ratio test (LRT)

Generally, a better test than the Wald is a LRT. The LRT statistic is again:



To perform a test of H0: βr = 0 vs. Ha: βr ≠ 0, we obtain the estimated probabilities of success from estimating



(suppose the estimates are , , … ) and the estimated probabilities of success from estimating



(suppose the estimates are , , … ). We can then find





 (1)

where I use  and  to be vectors of parameters under the H0 and Ha models. If the null hypothesis was true, -2log(Λ) has an approximate  distribution for a large sample.

Questions:

* What happens to Λ and -2log(Λ) as evidence grows against the null hypothesis?
* When should the null hypothesis be rejected?
* How should we calculate the p-value?

The hypotheses can be generalized to allow for q different β’s in H0 to be set to 0. When this happens and the null hypothesis is true, -2log(Λ) has an approximate  distribution for a large sample.

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

Consider the model with both change and distance included in it:



To perform Wald tests on each of these variables, we can examine the summary(mod.fit2) output from earlier:

> mod.fit2<-glm(formula = good ~ change + distance, family

 = binomial(link = logit), data = placekick)

> summary(mod.fit2)

Call: glm(formula = good ~ change + distance, family = binomial(link = logit), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

 -2.7061 0.2282 0.2282 0.3750 1.5649

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 5.893181 0.333184 17.687 <2e-16 \*\*\*

change -0.447783 0.193673 -2.312 0.0208 \*

distance -0.112889 0.008444 -13.370 <2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.4 on 1424 degrees of freedom

Residual deviance: 770.5 on 1422 degrees of freedom

AIC: 776.5

Number of Fisher Scoring iterations: 6

Below is a summary of the tests using α = 0.05:

|  |  |
| --- | --- |
| **Change** | **Distance** |
| H0: β1 = 0 Ha: β1 ≠ 0  | H0: β2 = 0 Ha: β2 ≠ 0  |
| Z0 = -2.31 | Z0 = -13.37 |
| p-value = 0.0208 | p-value < 2×10-16 |
| Reject H0 because p-value is small | Reject H0 because p-value is small |
| There is sufficient evidence to indicate change has an effect on the probability of success given distance is in the model. | There is sufficient evidence to indicate distance has an effect on the probability of success given change is in the model. |

Change has a p-value less than 0.05, but it is only a little less. In fact, if α = 0.01, there would not be a rejection of the null hypothesis. It is preferable to word a conclusion like

There is **marginal** evidence to indicate that change has an effect on the probability of success given distance is in the model.

There are a number of ways to perform LRTs in R. The easiest way to perform the tests of interest here is to use the Anova() function from the car package. This package is not automatically installed in R so you will need to install it prior to its use. The package corresponds to the book “An R Companion to Applied Regression” by Fox and Weisberg. Below is the R code and output:

> library(package = car)

> Anova(mod = mod.fit2, test = "LR")

Analysis of Deviance Table (Type II tests)

Response: good

 LR Chisq Df Pr(>Chisq)

change 5.246 1 0.02200 \*

distance 218.650 1 < 2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Notice the p-values are quite similar to the p-values for the Wald test. This is due to the large sample.

Within the stats package, the anova() function can perform LRTs. Below is what occurs with a somewhat naïve use of it:

> anova(object = mod.fit2, test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit

Response: good

Terms added sequentially (first to last)

 Df Deviance Resid. Df Resid. Dev P(>|Chi|)

NULL 1424 1013.43

change 1 24.277 1423 989.15 8.343e-07 \*\*\*

distance 1 218.650 1422 770.50 < 2.2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The p-value for distance is the same as from using Anova(), but the p-value for change is not. The reason for the difference is due to the hypotheses being tested. The anova() function tests the model’s explanatory variables in an sequential manner. Thus, the change test p-value is actually for the test of



because it is listed first in the formula argument of glm(). The distance variable is listed second so anova() tests:



where change is assumed to be in both models.

In order to produce the tests like Anova(), we need to estimate the H0 and Ha models separately and then use their model fit objects in a different way with anova():

> mod.fit.Ho<-glm(formula = good ~ distance, family =

 binomial(link = logit), data = placekick)

> anova(mod.fit.Ho, mod.fit2, test = “Chisq”)

Analysis of Deviance Table

Model 1: good ~ distance

Model 2: good ~ change + distance

 Resid. Df Resid. Dev Df Deviance P(>|Chi|)

1 1423 775.75

2 1422 770.50 1 5.2455 0.02200 \*

Signif. Codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ‘ 1

This use of anova() helps to emphasize a reduced and full model approach to obtaining the -2log(Λ) statistic. Also, this approach is helpful to know how to do when Anova() may not be available (with more complex models than logistic regression) or when more than one variable is being tested at a time.

Question: Why do you think these function names involve the word “anova” when Analysis of Variance is not being performed?

Of course, you could program into R the -2log(Λ) statistic yourself and not even use Anova() or anova(). Below is an example that tests



> mod.fit.Ho<-glm(formula = good ~ 1, family =

 binomial(link = logit), data = placekick)

> mod.fit.Ha<-glm(formula = good ~ change, family =

 binomial(link = logit), data = placekick)

> anova(mod.fit.Ho, mod.fit.Ha, test = "Chisq")

Analysis of Deviance Table

Model 1: good ~ 1

Model 2: good ~ change

 Resid. Df Resid. Dev Df Deviance P(>|Chi|)

1 1424 1013.43

2 1423 989.15 1 24.277 8.343e-07 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

> pi.hat.Ho<-mod.fit.Ho$fitted.values

> pi.hat.Ha<-mod.fit.Ha$fitted.values

> y<-placekick$good

> stat<--2\*sum(y\*log(pi.hat.Ho/pi.hat.Ha) + (1-y)\*log((1-

 pi.hat.Ho)/(1-pi.hat.Ha)))

[1] 24.27703

> pvalue<-1-pchisq(q = stat, df = 1)

> data.frame(stat, pvalue)

 stat pvalue

1 24.27703 8.342812e-07

Again, knowing how to perform a test in this manner is useful when there are no Anova() or anova() functions available – perhaps for a statistical research problem! Also, going through calculations like this is helpful to understand the test statistic being used.

Deviance

In the previous example's output, the word “deviance” and its abbreviation “dev” appeared a number of times. Deviance refers to the amount that a particular model deviates from another model as measured by -2log(Λ). For example, the -2log(Λ) = 5.246 value used for testing the change variable in the last example (given distance is in the model) is a measure of how much the estimated probabilities of success for the null hypothesis model deviate from the estimated probabilities of success for the alternative hypothesis model.

Residual deviance – This denotes how much the null hypothesis model of interest deviates from using the observed proportion of successes for each “observation” (yi = 0 or 1 for Bernoulli response data or wj/nj for binomial response data) to estimate π (πi or πj).

Using the observed proportion of successes is equivalent to using a model of the form  (i = 1, …, n) for Bernoulli response data or of the form  (j = 1, …, J) for binomial response data. Because the number of parameters is equal to the number of observations, this model is frequently referred to as the saturated model because no more additional parameters can be estimated.

Null deviance – This denotes how much the model  (or ) deviates from using the observed proportion of successes for each observation to estimate πi or (πj). Because  contains only the intercept term (just one parameter), every πi is estimated to be the same value for this particular model.

Deviance statistics are often calculated as an intermediary step for performing a LRT to compare two models. For example, consider testing:



The residual deviance for the model tests



Ha:Saturated model

with

 (2)

Note: If written in terms of binomial response data, we would have:



The residual deviance for the model  tests



Ha:Saturated model

with

 (3)

By finding (2) – (3) we obtain Equation (1); thus, we obtain the desired -2log(Λ) by subtracting the residual deviances of the models in our original hypotheses!

The degrees of freedom for the test can be found from subtracting the corresponding degrees of freedom for (3) from (2).

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

Below is the R code demonstrating the above discussion with respect to



> mod.fit.Ho<-glm(formula = good ~ distance, family =

 binomial(link = logit), data = placekick)

> df<-mod.fit.Ho$df.residual-mod.fit2$df.residual

> stat<-mod.fit.Ho$deviance-mod.fit2$deviance

> pvalue<-1-pchisq(q = stat, df = df)

> data.frame(Ho.resid.dev = mod.fit.Ho$deviance,

 Ha.resid.dev = mod.fit2$deviance, df = df, stat =

 round(stat,4), pvalue = round(pvalue,4))

 Ho.resid.dev Ha.resid.dev df stat pvalue

1 775.745 770.4995 1 5.2455 0.022

Final note for this section: For tests and projects, use the easiest ways to perform the LRTs unless asked to otherwise. Some of the longer ways to perform the LRT were shown to:

1. Reinforce how the statistic was found,
2. Prepare you for material later in the course, and
3. Show common ways that others may perform tests.

**Section 2.2.3 – Odds ratios**

A logistic regression model can be written as



where the left-side of the model is the log odds of a success. Using a similar interpretation as for normal linear regression models, we can look at βr then to interpret the effect that xr has on this log odds of a success. We can then form odds ratios by looking at these odds of success at different values of xr.

For ease of presentation, consider the logistic regression with only one explanatory variable x:

 

We can re-write this model as



where I replaced the π/(1 – π) to help with the notation. For example, the odds of a success are  for the placekicking example, where distance is the only explanatory variable and it is set to a value of 30 yards.

If we increase x by c-units, the odds of a success becomes



To interpret the effect of increasing x by c-units, we can form an odds ratio:



Notice that x falls out!!! Thus, it does not matter what the value of x is, the odds ratio remains the same for a c-unit increase. This is one of the main reasons why logistic regression is the most used way to model binary response data.

On your own, verify  for the model given at the beginning of this section.

There are a number of ways to interpret the odds ratio in the context of logistic regression. I recommend using the following:

The odds of a success change by  times for every c-unit increase in x.

This interpretation is worded a little different from what we had in Section 1.2. The reason is due to x not necessarily having ONLY two levels.

Suppose x did only have two levels coded as 0 or 1 as commonly done for indicator variables in normal linear regression. This leads to

 and 

as the only possible odds. The odds ratio becomes

 

In this situation, you could say

The odds of a success are  times as large for x = 1 than for x = 0.

To find the estimated odds ratio, simply replace the parameter with its corresponding estimate:



The interpretation of the odds ratio now needs to have an “estimated” inserted in the appropriate location. This estimate is the MLE.

Confidence intervals for OR

Wald confidence intervals are the easiest to calculate. First, an interval for cβ1 needs to be found:



where  is obtained from the estimated covariance matrix for the parameter estimates. Notice where c is located in the interval calculation. The second c comes about through .

You may have seen in another class that for a random variable Y and constant a, Var(aY) = a2Var(Y).

To find the (1 – α) Wald confidence interval for OR, we use the exponential function:



As you might expect, Wald confidence intervals do not always work as well as we would like for smaller sample sizes. Instead, a better interval is a profile likelihood ratio interval. For a (1 – α)100% interval, we find the set of β1 values such that



is satisfied. On the left-side, we have the usual -2log(Λ) form, but without a specified value of β1. The  is an estimate of β0 for a fixed value of β1. Iterative numerical procedures can be used to find the β1 values which satisfy the above equation.

An ad-hoc approach would be to simply try a large number of possible values of β1 and obtain  for each β1. Next, the -2log(Λ) equation would be calculated to see which β1’s satisfy . The smallest and largest β1 values that satisfy the inequality would be used as the lower and upper confidence interval limits. Please see my program for an example.

The (1 – α)100% interval for OR is then



using the lower and upper limits found for β1 in the above equation.

A standard interpretation for the interval is

With (1 – α)100% confidence, the odds of a success change by an amount between <lower limit> to <upper limit> times for every c-unit increase in x.

Comments about the use of odds ratios with logistic regression models:

1. In many instances, inverting odds ratios less than 1 is helpful for interpretation purposes.
2. An appropriate value of c should be chosen in the context of the explanatory variable. For example, if 0.1 < x < 0.2, a value of c = 1 would not be appropriate. Additionally, if 0 < x < 1000, a value of c = 1 may not be appropriate as well.
3. When there is more than one explanatory variable, the same interpretation of the odds ratio *generally* can be made with the addition of “holding the other explanatory variables constant” added. This is basically the same as what is done in normal linear regression.
4. If there are interaction, quadratic, or other transformations of the explanatory variables contained within the model, the odds ratio is not simply  as given previously, because the odds ratio is no longer constant for every c-unit increase in x. We will examine what the odds ratio value is later in Section 2.2.5.
5. A categorical explanatory variable represented by multiple indicator variables does not have the same type of interpretation as given previously. We will examine this further in Section 2.2.6.

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

Consider the model with only distance as the explanatory variable:



To estimate the odds ratio, we can simply use the exp() function:

> exp(mod.fit$coefficients[2])

 distance

 0.8913424

> 1/exp(10\*mod.fit$coefficients[2])

 distance

 3.159035

The first odds ratio is for a 1-yard (c = 1) increase in distance. This is not very meaningful in the context of the problem! Instead, c = 10 would be much more meaningful because 10 yards are needed for a first down in football. Also, it is more meaningful to look at a 10 yard decrease (another first down) rather than a 10 yard increase. Therefore, the estimated odds of a success change by 1/e10×(-0.1150) = 3.16 times for every 10 yard decrease in the distance of the placekick.

Because the odds of success are larger for a 10 yard decrease, a football coach would prefer to go for a field goal at the shorter distance (if possible). While this result is not surprising for football fans, this research quantifies the amount of benefits, which was previously unknown. Also, remember that the 3.16 holds for comparing 30 to 20-yard placekicks as well as 55 to 45-yard placekicks or any other 10-yard decrease.

To account for the variability in the odds ratio estimator, we would like to calculate a confidence interval for the actual odds ratio itself. Below is the code for the profile likelihood ratio interval:

> beta.ci<-confint(object = mod.fit, parm = "distance",

 level = 0.95)

Waiting for profiling to be done...

> beta.ci

 2.5 % 97.5 %

 -0.13181435 -0.09907104

> rev(1/exp(beta.ci\*10)) #Invert OR C.I. and c=10

 97.5 % 2.5 %

 2.693147 3.736478

> #Remove labels with as.numeric()

> as.numeric(rev(1/exp(beta.ci\*10)))

[1] 2.693147 3.736478

The confint() function first finds an interval for β1 itself. We then use the exp() function to find the confidence interval for OR. The 95% profile likelihood ratio confidence interval is 2.69 < OR < 3.74 – pay special attention to how this was found with the rev() function and the beta.ci object. The standard interpretation of the interval is

With 95% confidence, the odds of a success change by an amount between 2.69 to 3.74 times for every 10 yard decrease in the distance of the placekick.

Because the interval is entirely above 1, there is sufficient evidence that a 10 yard decrease in distance increases the odds of a successful placekick.

While a profile likelihood ratio interval is usually preferred, it is instructive to see how to calculate a Wald interval as well, because there will be instances later where we will not be able to calculate a profile likelihood ratio interval.

> beta.ci<-confint.default(object = mod.fit, parm =

 "distance", level = 0.95)

> beta.ci

 2.5 % 97.5 %

distance -0.1313709 -0.0986824

> rev(1/exp(beta.ci\*10)) #Invert OR C.I. with c=10

[1] 2.682701 3.719946

I used the confint.default() function for part of the calculations, because there is no “Wald” like option in confint(). The 95% Wald interval is 2.68 < OR < 3.72, which is similar to the profile likelihood ratio interval due to the large sample size.

To see how these calculations are performed without the confint.default() function, below is an example of how to program into R the corresponding formula:

> beta.ci<-mod.fit$coefficients[2] + qnorm(p = c(0.025,

 0.975))\*sqrt(vcov(mod.fit)[2,2])

> beta.ci

[1] -0.1313709 -0.0986824

> rev(1/exp(beta.ci\*10))

[1] 2.682701 3.719946

I show in the corresponding program how to find the profile likelihood ratio interval without using confint(). It is instructive to see how this done because it shows how to use the uniroot() function and how to estimate a logistic regression model with a fixed constant in the formula argument of glm().

**Section 2.2.4 – Probability of success**

As shown earlier, the estimate for π is



To find a confidence interval for π, consider again 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 (1 – α)100% 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.

The mcprofile package (not in the default installation of R) provides a general way to compute profile likelihood ratio intervals. Earlier versions of it produced questionable results at times. Current versions generally do not have problems. Still, I recommend using the following approach with this package:

1. Calculate a Wald interval.
2. Calculate a profile likelihood ratio interval with the mcprofile package.
3. Use the profile LR interval as long as it is not outlandishly different than the Wald and there are no warning messages given by R when calculating the interval. Otherwise, use the Wald interval.

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.

Wald interval

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

Note that 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, ).

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])

 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. Please make sure that you could also calculate this by directly coding the formulas given earlier (see example in program).

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 likelihood ratio interval:

> library(package = mcprofile)

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

> K

 [,1] [,2]

[1,] 1 20

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

 #Calculate -2log(Lambda)

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

 0.95) #CI for beta\_0 + beta\_1 \* x

> 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 .
2. 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.
3. The confint() function uses all of these values to find the 95% confidence interval for β0 + β120.
4. 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)

Next, let’s look at a visual assessment of the logistic regression model when there is only one explanatory variable x. For a normal linear model, we would examine something like



The closeness of the points to the estimated regression model line tell us something about how well an observation is fit by the model.

Because the response variable in logistic regression is binary, constructing a plot like this is not very informative because all plotted points would be at y = 0 or 1 on the y-axis. However, we can plot the observed proportion of successes at each x instead to obtain a general understanding of how well the model fits the data.

This only works well when the number of observations at each possible x is not small. In the case of a truly continuous x, the number of observations would be 1 and this plot generally would not be very useful. Alternatives for this situation include grouping observations by x and finding the observed proportion of successes for each group; however this leads to potentially different results depending on how the grouping is done

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

Earlier we found the number of observations and number of successes at each unique distance:

> w<-aggregate(formula = good ~ distance, data = placekick,

 FUN = sum)

> n<-aggregate(formula = good ~ distance, data = placekick,

 FUN = length)

> w.n<-data.frame(distance = w$distance, success = w$good,

 trials = n$good, proportion = round(w$good/n$good,4))

> head(w.n)

 distance success trials proportion

1 18 2 3 0.6667

2 19 7 7 1.0000

3 20 776 789 0.9835

4 21 19 20 0.9500

5 22 12 14 0.8571

6 23 26 27 0.9630

This was used to estimate a logistic regression model using a binomial response form of the data. Instead, we can plot the observed proportion of successes at each distance and overlay the estimated logistic regression model:

> win.graph(width = 7, height = 6, pointsize = 12)

> plot(x = w$distance, y = w$good/n$good, xlab="Distance

 (yards)", ylab="Estimated probability", panel.first =

 grid(col = "gray", lty = "dotted"))

> curve(expr = predict(object = mod.fit, newdata =

 data.frame(distance = x), type = "response"), col =

 "red", add = TRUE, xlim = c(18, 66))



Previously, I used the curve function with

> curve(expr = exp(mod.fit$coefficients[1] +

 mod.fit$coefficients[2]\*x) / (1 +

 exp(mod.fit$coefficients[1] +

 mod.fit$coefficients[2]\*x)), col = "red", xlim = c(18,

 66), ylab = expression(hat(pi)), xlab = "Distance",

 main = "Estimated probability of success for a

 placekick", panel.first = grid())

to plot the model. Now, I use a simpler way with the predict() function.

Question: If we were to assess the fit of the logistic regression model here in the same manner as for a normal linear regression model, what would you conclude?

To include a measure of how many observations are at each distance, we can use a bubble plot. For this plot, we are going to make the plotting point size proportional to the observed number of observations at each unique distance.

> win.graph(width = 7, height = 6, pointsize = 12)

> symbols(x = w$distance, y = w$good/n$good, circles =

 sqrt(n$good), inches = 0.5, xlab="Distance (yards)",

 ylab="Estimated probability", panel.first = grid(col =

 "gray", lty = "dotted"))

> curve(expr = predict(object = mod.fit, newdata =

 data.frame(distance = x), type = "response"), col =

 "red", add = TRUE, xlim = c(18, 66))



The circles argument specifies how to scale the plotting points. I used the sqrt() function within this argument value only to lessen the absolute differences between the values in n$good, and this does not need to be used in general.

There are 789 observations at a distance of 20 yards. The next largest number of observations is 30, which is at a distance of 32 yards. Due to the large absolute difference in the number of observations, this causes the plotting point at 20 yards to be very large and the other plotting points to be very small.

Points that may have caused us concern before, now generally do not because we see they represent a small number of observations. For example,

* 18-yard placekicks: Only 3 observations occurred and there were 2 successes.
* The largest in distance placekicks: Many of these correspond to 1 observation only.

However, there are a few large plotting points, such as at 32 yards ( = 0.89, observed proportion = 23/30 = 0.77) and 51 yards ( = 0.49, observed proportion = 11/15 = 0.73), that may not be fit well by the model. How to more formally assess these observations and others will be an important subject of Chapter 5 when we examine model diagnostic measures.

Question: Suppose the previous plot looked like this:



What do you think about the fit of the model?

Another commonly made plot in normal linear regression with one explanatory variable is a plot of the estimated regression model with confidence interval bands:



We can add these bands to the previous plots for the estimated logistic regression models.

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

I wrote a function called ci.pi() to help automate some of the needed calculations:

> ci.pi<-function(newdata, mod.fit.obj, alpha){

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

 newdata, type = "link", se = TRUE)

 CI.lin.pred.lower<-linear.pred$fit - qnorm(p = 1-

 alpha/2)\*linear.pred$se

 CI.lin.pred.upper<-linear.pred$fit + qnorm(p = 1-

 alpha/2)\*linear.pred$se

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

 exp(CI.lin.pred.lower))

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

 exp(CI.lin.pred.upper))

 list(lower = CI.pi.lower, upper = CI.pi.upper)

}

> #Test case

> ci.pi(newdata = data.frame(distance = 20), mod.fit.obj =

 mod.fit, alpha = 0.05)

$lower

 1

0.9597647

$upper

 1

0.9791871

**Note that you will find this function quite useful for your own calculations whenever you need a confidence interval for π!**

To add the confidence interval bands to the previous plot, we can use the following code:

> curve(expr = ci.pi(newdata = data.frame(distance = x),

 mod.fit.obj = mod.fit, alpha = 0.05)$lower, col =

 "blue", lty = "dotdash", add = TRUE, xlim = c(18, 66))

> curve(expr = ci.pi(newdata = data.frame(distance = x),

 mod.fit.obj = mod.fit, alpha = 0.05)$upper, col =

 "blue", lty = "dotdash", add = TRUE, xlim = c(18, 66))

> legend(x = 20, y = 0.4, legend = c("Logistic regression

 model", "95% individual C.I."), lty = c("solid",

 "dotdash"), col = c("red", "blue"), bty = "n")



Therefore, we have the 95% confidence interval at each possible distance. Note that the familywise error rate though is not controlled.

**Section 2.2.5 – Interactions and transformations for explanatory variables**

Just as in normal linear regression models, we can include various transformations of explanatory variables in a logistic regression model. I will focus on the most common: 1) two-way interactions and 2) quadratic terms.

Interactions

Interactions between explanatory variables are needed when the effect of one explanatory variable on the probability of success depends on the value for a second explanatory variable.

Consider the model of



Below are three ways this model can be coded into the formula argument of glm() when there are two variables called x1 and x2 in a data frame:

* formula = y ~ x1 + x2 + x1:x2
* formula = y ~ x1\*x2
* formula = y ~ (x1 + x2)^2

The colon indicates an interaction term between variables. The asterisk indicates all interactions AND lower order effects (only x1 and x2). The ^2 indicates all two-way interactions and lower order effects.

Consider the model of



Below are three ways this model can be coded into the formula argument of glm() when there are two variables called x1, x2, and x3 in a data frame:

* formula = y ~ x1 + x2 + x3 + x1:x2 +
 x1:x3 + x2:x3
* formula = y ~ x1\*x2 + x1\*x3 + x2\*x3
* formula = y ~ (x1 + x2 + x3)^2

Comments:

1. The individual explanatory variables are often referred to as “main effect” terms in a model.
2. I will use the convention of including all lower order interactions and main effects whenever an interaction is included. For example, if a three-way interaction is included like x1x2x3, all two-way interactions and main effects among the explanatory variables would also be included.

Consider again the model of



The effect that x1 has on logit(π) is dependent on the specific value of x2. Thus, we no longer can only look at β1 when trying to understand the effect x1 has on the response. Similarly, the effect that x2 has on logit(π) is dependent on the specific value of x1. While we can still use odds ratios to interpret these effects, it is now a little more difficult. However, please remember that the interpretation still is based on the ratio of two odds.

For the above model, the odds ratio for x2 holding x1 constant is



Notice the role that x1 plays in OR. Thus, you will always need to include x1 when interpreting x2’s corresponding odds ratio:

The odds of a success change by  times for every c-unit increase in x2 when x1 is fixed at a value of \_\_\_.

Compare this to the odds ratio interpretation for x2 for the model of 

The odds of a success change by  times for every c-unit increase in x2 holding x1 constant.

Wald and profile likelihood ratio intervals again can be found for OR. With respect to the Wald interval, we use the same basic form as before, but now with a more complicated variance expression. For example, the interval for the x2 odds ratio in the  model is:



where



Of course, the variances and covariances can be found from the estimated covariance matrix.

Profile likelihood ratio intervals are generally preferred. However, they can be more difficult to calculate due to the additional parameters included in the odds ratio similar to what we saw in Section 2.2.4. As in the previous section, I recommend always calculating both Wald and profile likelihood ratio intervals. If there appears to be problems with using the mcprofile package, use the Wald interval.

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

Generally, one would conjecture that the more time that a football is in the air (more time equates to a longer distance), the more it is susceptible to the effect of wind.

For example, a 50-yard placekick will have a longer time period that the wind can affect it than a 20-yard placekick.

Thus, a distance and wind interaction would be of interest to test. The wind explanatory variable in the data set is a binary variable for placekicks attempted in windy conditions (1) vs. non-windy conditions (0) placekicks, where windy conditions are defined as a wind stronger than 15 miles per hour at kickoff in an outdoor stadium.

Below is how a model including distance, wind, and the distance×wind interaction is estimated:

> mod.fit.Ha<-glm(formula = good ~ distance + wind +

 distance:wind, family = binomial(link = logit), data =

 placekick)

> summary(mod.fit.Ha)

Call: glm(formula = good ~ distance + wind + distance:wind, family = binomial(link = logit), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

-2.7291 0.2465 0.2465 0.3791 1.8647

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 5.684181 0.335962 16.919 <2e-16 \*\*\*

distance -0.110253 0.008603 -12.816 <2e-16 \*\*\*

wind 2.469975 1.662144 1.486 0.1373

distance:wind -0.083735 0.043301 -1.934 0.0531 .

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1013.43 on 1424 degrees of freedom

Residual deviance: 767.42 on 1421 degrees of freedom

AIC: 775.42

Number of Fisher Scoring iterations: 6

The estimated logistic regression model is



Wald test information for the test of H0: β3 = 0 vs. Ha: β3 ≠ 0 is given in the output. The test statistic is Z0 = -1.934, and the p-value is 0.0531. Thus, there is marginal evidence of a distance and wind interaction.

To perform a LRT, we can fit the model under the null hypothesis and then use the anova() function:

> mod.fit.Ho<-glm(formula = good ~ distance + wind, family

 = binomial(link = logit), data = placekick)

> anova(mod.fit.Ho, mod.fit.Ha, test = "Chisq")

Analysis of Deviance Table

Model 1: good ~ distance + wind

Model 2: good ~ distance + wind + distance:wind

 Resid. Df Resid. Dev Df Deviance P(>|Chi|)

1 1422 772.53

2 1421 767.42 1 5.1097 0.02379 \*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The test statistic is -2log(Λ) = 5.1097, and the p-value is 0.0238. Again, there is marginal evidence of a distance and wind interaction.

If I used a strict α = 0.05 type I error rate, what would have been the conclusions?

Notes:

* The negative coefficient on the distance main effect indicates that when wind = 0, the probability of success decreases with increasing distance. The negative coefficient on the interaction indicates that this effect is worsened under windy conditions.
* Another way to obtain the LRT information would be to use the Anova() function from the car package:

> Anova(mod.fit.Ha, test = "LR")

Analysis of Deviance Table (Type II tests)

Response: good

 LR Chisq Df Pr(>Chisq)

distance 238.053 1 < 2e-16 \*\*\*

wind 3.212 1 0.07312 .

distance:wind 5.110 1 0.02379 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

What is the p-value for wind testing?

* A graph of the estimated logistic regression model is quite useful to see how windy conditions affect the probability of success. Using the curve() function in a similar manner as before, below are plots of the models with and without the interaction:



Odds ratios

To begin using odds ratios in this setting, we need to find the odds ratio for wind comparing windy (1) vs. non-windy (0) holding distance constant. Below is the formula from our earlier discussion:



For this equation, x1 is distance and x2 is wind. Of course, c = 1 for this setting due to wind being binary. Because distance could be anywhere from 18 to 66 yards, I will choose distances of 20, 30, 40, 50, and 60 yards to interpret the odds ratio for wind. Below is the code:

> beta.hat<-mod.fit.Ha$coefficients[2:4]

> c<-1

> distance<-seq(from = 20, to = 60, by = 10)

> OR.wind<-exp(c\*(beta.hat[2] + beta.hat[3]\*distance))

> cov.mat<-vcov(mod.fit.Ha)[2:4,2:4]

> #Var(beta^\_2 + distance\*beta^\_3)

> var.log.OR<-cov.mat[2,2] + distance^2\*cov.mat[3,3] +

 2\*distance\*cov.mat[2,3]

> ci.log.OR.low<-c\*(beta.hat[2] + beta.hat[3]\*distance) –

 c\*qnorm(p = 0.975)\*sqrt(var.log.OR)

> ci.log.OR.up<-c\*(beta.hat[2] + beta.hat[3]\*distance) +

 c\*qnorm(p = 0.975)\*sqrt(var.log.OR)

> round(data.frame(distance = distance, OR.hat = 1/OR.wind,

 OR.low = 1/exp(ci.log.OR.up), OR.up =

 1/exp(ci.log.OR.low)),2)

 distance OR.hat OR.low OR.up

1 20 0.45 0.09 2.34

2 30 1.04 0.40 2.71

3 40 2.41 1.14 5.08

4 50 5.57 1.54 20.06

1. 60 12.86 1.67 99.13

Note that I put the estimated ’s into one object so that I could use the same [index] values as would correspond to , , and  in my model (remember that  is the [1] element of mod.fit.Ha$coefficients).

As originally conjectured, the longer the distance, the more it is susceptible to the wind. Notice that from 40 yards on, the confidence interval does not contain 1. Thus, there is sufficient evidence at those distances to indicate wind has an effect on the success or failure of the placekick.

Example interpretation: With 95% confidence, the odds of a success are between 1.14 to 5.08 times as large for windy (wind = 1) than for non-windy (wind = 0) placekicks at a distance of 40 yards.

For the distance odds ratio, we need to hold wind constant at 0 or 1. Also, we need to choose a value for c with respect to distance. The OR equation is



Below is my code:

> c<-10 #10-yard increment

> wind<-0:1 #Examine wind for 0 and 1

> OR.dist<-exp(c\*(beta.hat[1] + beta.hat[3]\*wind))

 #Estimated OR

> var.log.OR<-cov.mat[1,1] + wind^2\*cov.mat[3,3] +

 2\*wind\*cov.mat[1,3] #Var(beta^\_2 + distance\*beta^\_3)

> ci.log.OR.low<-c\*(beta.hat[1] + beta.hat[3]\*wind) –

 c\*qnorm(p = 0.975)\*sqrt(var.log.OR)

> ci.log.OR.up<-c\*(beta.hat[1] + beta.hat[3]\*wind) +

 c\*qnorm(p = 0.975)\*sqrt(var.log.OR)

> data.frame(OR.dist, OR.low = exp(ci.log.OR.low), OR.up =

 exp(ci.log.OR.up))

 OR.dist OR.low OR.up

1 0.3320311 0.28051271 0.3930113

2 0.1437214 0.06255906 0.3301814

> round(data.frame(wind = wind, OR.hat = 1/OR.dist, OR.low

 = 1/exp(ci.log.OR.up), OR.up = 1/exp(ci.log.OR.low)),2)

 #Inverted

 wind OR.hat OR.low OR.up

1 0 3.01 2.54 3.56

2 1 6.96 3.03 15.98

Notice the odds ratios are inverted. Below are the interpretations:

* With 95% confidence, the odds of a success change by an amount between 2.54 to 3.56 times for every 10-yard decrease in distance under non-windy conditions.
* With 95% confidence, the odds of a success change by an amount between 3.03 to 15.98 times for every 10-yard decrease in distance under windy conditions.

I think football fans would be more interested in the first set of odds ratios where we conditioned on the distance value.

Below is my code for the profile LR intervals:

> library(package = mcprofile) # Need if had not already

 been used

> K<-matrix(data = c(0, 0, 1, 20,

 0, 0, 1, 30,

 0, 0, 1, 40,

 0, 0, 1, 50,

 0, 0, 1, 60), nrow = 5, ncol = 4,

 byrow = TRUE)

> K

 [,1] [,2] [,3] [,4]

[1,] 0 0 1 20

[2,] 0 0 1 30

[3,] 0 0 1 40

[4,] 0 0 1 50

[5,] 0 0 1 60

> # A little quicker way to form K

> # distance<-seq(from = 20, to = 60, by = 10)

> # K<-cbind(0, 0, 1, distance)

> # profile LR

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

> ci.log.OR<-confint(object = linear.combo, level = 0.95,

 adjust = "none")

> # ci.log.OR

> data.frame(distance, OR.hat =

 round(1/exp(ci.log.OR$estimate), 2),

 OR.low = round(1/exp(ci.log.OR$confint$upper), 2),

 OR.up = round(1/exp(ci.log.OR$confint$lower), 2))

 distance Estimate OR.low OR.up

C1 20 0.45 0.06 1.83

C2 30 1.04 0.34 2.42

C3 40 2.41 1.14 5.15

C4 50 5.57 1.68 22.79

C5 60 12.86 2.01 130.85

With the matrix formulation given in K, you can use the wald() function in the mcprofile package for a more convenient way to find multiple Wald intervals:

> save.wald<-wald(object = linear.combo)

> ci.log.OR.wald<-confint(object = save.wald, level = 0.95,

 adjust = "none")

> data.frame(distance, OR.hat =

 round(1/exp(ci.log.OR.wald$estimate), 2),

 OR.low = round(1/exp(ci.log.OR.wald$confint$upper), 2),

 OR.up = round(1/exp(ci.log.OR.wald$confint$lower), 2))

 distance Estimate OR.low OR.up

C1 20 0.45 0.09 2.34

C2 30 1.04 0.40 2.71

C3 40 2.41 1.14 5.08

C4 50 5.57 1.54 20.06

C5 60 12.86 1.67 99.13

A similar function exists in the multcomp package. This package is used for estimating contrasts and performing multiple comparisons in a number of settings (there is an exercise involving it in my book).

My program contains the profile LR interval code for an odds ratio with a 10-yard decrease in distance holding wind constant at 0 or 1.

Question: Could confint() be used to find any of the profile LR interval? If so, which ones?

Quadratic terms

Quadratic and higher order polynomials are needed when the relationship between an explanatory variable and logit(π) is not linear. To include this type of transformation in a formula argument, we can use the carat symbol ^ with the degree of the polynomial. However, as we saw earlier in this section, the carat symbol is used to denote the order of the interaction between explanatory variables. Thus,

formula = y ~ x1 + x1^2

would NOT be interpreted as . R interprets the x1^2 part as “all two-way interactions involving x1.” Because only one explanatory variable is given in x1^2, R interprets this as simply x1. Also, because x1 was already given in the formula argument, the variable is not duplicated, so  would be the model estimated.

In order to obtain a  term, we need to use the I() function with x1^2. The I() function instructs R to interpret arguments as it normally would. Thus,

formula = y ~ x1 + I(x1^2)

would be interpreted as .

Question: How would you code the model



into a formula argument?

Odds ratios involving polynomial terms are dependent on the explanatory variable of interest. For example, to find OR for x1 in , we form the odds ratio as:



The standard interpretation becomes

The odds of a success change by  times for a c-unit increase in x1 when x1 is at a value of \_\_\_.

Because the odds ratio is dependent on the explanatory variable value, it is better to change the interpretation to

The odds of a success are  times as large for x1 = \_\_ + c than for x1 = \_\_,

where you need to put in the appropriate value of x1. Also, this means multiple odds ratios may be needed to fully understand the effect of x1 on the response.

Wald confidence intervals are found in a similar manner as for interaction terms. For the model of , the interval is



where



Profile likelihood ratio intervals can be calculated as well, but they are subject to the same problems as before with the mcprofile package.

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

Suppose x represents the distance of the placekick. Below is how we can estimate the model



using glm():

> mod.fit.distsq<-glm(formula = good ~ distance +

 I(distance^2), family = binomial(link = logit), data =

 placekick)

> summary(mod.fit.distsq)

Call:

glm(formula = good ~ distance + I(distance^2), family = binomial(link = logit),

 data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

-2.8625 0.2175 0.2175 0.4011 1.2865

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 7.8446831 1.0009079 7.838 4.59e-15 \*\*\*

distance -0.2407073 0.0579403 -4.154 3.26e-05 \*\*\*

I(distance^2) 0.0017536 0.0007927 2.212 0.027 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.43 on 1424 degrees of freedom

Residual deviance: 770.95 on 1422 degrees of freedom

AIC: 776.95

Number of Fisher Scoring iterations: 6

The estimated models is



The p-value for the Wald test of H0:β2 = 0 vs. Ha: β2 ≠ 0 is 0.027 suggesting there is marginal evidence of a quadratic relationship between distance and the response. A LRT provides a similar p-value:

> library(package = car)

> Anova(mod.fit.distsq)

Analysis of Deviance Table (Type II tests)

Response: good

 LR Chisq Df Pr(>Chisq)

distance 16.9246 1 3.889e-05 \*\*\*

I(distance^2) 4.7904 1 0.02862 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Below is a plot of the estimated model, where I also include the estimate of :



The main difference between the two models appears to be for the larger distances. Given the small number of observations at those distances, it may be difficult to justify the need for the quadratic term. We will investigate this further in Chapter 5 when performing model building to find the best overall model for this data set.

Please investigate on your own how to calculate the odds ratios. Some example code is given in the corresponding program to help you.

**Section 2.2.6 – Categorical explanatory variables**

As we saw earlier with the change variable in the placekicking data set, a categorical explanatory variable with two levels can be represented simply as a binary variable to reflect these two levels. When there are q levels (where q > 2), only q – 1 binary (often referred to as “indicator”) variables are needed to represent the variable – just like in normal linear regression. If you would like to review this in a normal linear regression model context, please see my STAT 870 lecture notes on this topic (Section 8.3 of Chapter 8 at [www.chrisbilder.com/stat870/schedule.htm](http://www.chrisbilder.com/stat870/schedule.htm)).

Example: Categorical explanatory variable with 4 levels (4levels.R)

Suppose an explanatory variable has levels of A, B, C, and D. Three indicator variables can be used to represent the explanatory variable in a model:

|  |  |
| --- | --- |
|  | Indicator variables |
| Levels | x1 | x2 | x3 |
| A | 0 | 0 | 0 |
| B | 1 | 0 | 0 |
| C | 0 | 1 | 0 |
| D | 0 | 0 | 1 |

Notice how each level of the explanatory variable has a unique coding. The logistic regression model is



Informally, we could also write out the full model as



where it is assumed that B, C, or D are corresponding indicator variables in the model.

For example, A is the “base” level, and it is represented in the model with x1 = x2 = x3 = 0 so that



For category B, the model becomes



Thus, β1 measures the effect of level B when compared to level A.

R’s treatment of categorical explanatory variables

R treats categorical variables as a *factor* class type. By default, R orders the levels within a factor alphabetically, where numbers are ordered before letters and uppercase letters are before lower case letters:

 0, 1, 2, …, 9, …, a, A, b, B, …, z, Z

To see the ordering of any factor, the levels() function can be used. This ordering of levels is important because R uses it to construct indicator variables with the “set first level to 0” method of construction. This is what I used previously in the 4 level categorical example. The “set last level to 0” construction is used by some other software packages, like SAS.

Example: Categorical explanatory variable with 4 levels (4levels.R)

Below is a simple data set to help demonstrate R’s coding of the levels:

> set1<-data.frame(cat.var = c("D", "A", "A", "B", "D",

 "C"))

> set1

 cat.var

1 D

2 A

3 A

4 B

5 D

6 C

> class(set1$cat.var)

[1] "factor"

> levels(set1$cat.var)

[1] "A" "B" "C" "D"

> contrasts(set1$cat.var)

 B C D

A 0 0 0

B 1 0 0

C 0 1 0

D 0 0 1

The relevel() function in R can be used to define a new base level if desired:

> cat.var2<-relevel(x = set1$cat.var, ref = "D")

> set2<-data.frame(set1, cat.var2)

> set2

 cat.var cat.var2

1 D D

2 A A

3 A A

4 B B

5 D D

6 C C

> #A little quicker way

> set1$cat.var2<-relevel(x = set1$cat.var, ref = "D")

> set1

 cat.var cat.var2

1 D D

2 A A

3 A A

4 B B

5 D D

6 C C

> levels(set1$cat.var2)

[1] "D" "A" "B" "C"

> contrasts(set1$cat.var2)

 A B C

D 0 0 0

A 1 0 0

B 0 1 0

C 0 0 1

A more general function to reset the ordering of lebels is the factor() function:

> set1$cat.var3<-factor(set1$cat.var, levels = c("D", "A",

 "B", "C"))

> set1

 cat.var cat.var2 cat.var3

1 D D D

2 A A A

3 A A A

4 B B B

5 D D D

6 C C C

Additional comments:

* Because change in the placekicking data set was already coded as 0 or 1, R just simply treats it as a numerical variable rather than a factor.
* There are times when a categorical variable is coded numerically. For example, suppose a variable named “x” has levels of 1, 2, and 3 where 3 is not necessarily “greater than” 1 in its meaning. R will treat this by default as a numerical variable when estimating a model. To have R create indicator variables for x, either a) use factor(x) in a formula argument or create a new factor in the data frame with the factor() function.
* R will use a different ordering for a factor’s levels only when set through code. A perhaps unexpected occurrence of this is shown with the Salk vaccine clinical trial data in the homework.

Example: Control of the Tomato Spotted Wilt Virus (TomatoVirus.R, TomatoVirus.csv)

Plant viruses are often spread by insects. This occurs by insects feeding on plants already infected with a virus, and subsequently becoming carriers of the virus. When they feed on other plants, insects may transmit this virus back to these new plants.

To better understand the Tomato Spotted Wilt Virus and how to control thrips that spread it, researchers at Kansas State University performed an experiment in a number of greenhouses. One hundred uninfected tomato plants were put into each greenhouse, and they were introduced to the virus (“infested”) in one of two ways (coded levels of the corresponding variable are given in parentheses):

1. Interspersing additional infected plants among the clean ones, and then releasing "uninfected" thrips to spread the virus (1)
2. Releasing thrips that carry the virus (2)

To control the spread of the virus to the plants, the researchers used one of three methods:

1. Biologically through using predatory spider mites (B)
2. Chemically using a pesticide (C)
3. None (N)

The number of plants not displaying symptoms of infection were recorded for each greenhouse after 8 weeks. Below is the data where each row of the data set represents a greenhouse:

> tomato <- read.csv(file = "C:\\data\\TomatoVirus.csv")

> head(tomato)

 Infest Control Plants Virus8

1 1 C 100 21

2 2 C 100 10

3 1 B 100 19

4 1 N 100 40

5 2 C 100 30

6 2 B 100 30

From row 1 of the data set, we see that 21 out of 100 originally uninfected plants in a greenhouse showed symptoms after 8 weeks, and these plants had infestation method #1 applied while trying to control the thrips with a chemical application.

Both the Control and Infest explanatory variables are categorical in nature. Below is how R treats the Control variable:

> class(tomato$Control)

[1] "factor"

> levels(tomato$Control)

[1] "B" "C" "N"

> contrasts(tomato$Control)

 C N

B 0 0

C 1 0

N 0 1

Thus, the B level of Control is the base level, and there are two indicator variables for representing levels C and N. As for the Infest variable, notice it is coded numerically. Because the variable has only two levels, this will not affect our use of it in R. However, if it had more than two levels, we would need to transform the class of it to a factor through using the factor() function. Below is the corresponding code:

> class(tomato$Infest)

[1] "numeric"

> levels(tomato$Infest)

NULL

> class(factor(tomato$Infest))

[1] "factor"

> levels(factor(tomato$Infest))

[1] "1" "2"

> contrasts(factor(tomato$Infest))

 2

1 0

2 1

For demonstration purposes, I will change the Infest variable to be a factor in the data frame:

> tomato$Infest<-factor(tomato$Infest)

> class(tomato$Infest)

[1] "factor"

and use this for the remainder of the data analysis.

We estimate the model with both the Infest and Control variables in it:

> mod.fit<-glm(formula = Virus8/Plants ~ Infest +

 Control, family = binomial(link = logit), data =

 tomato, weight = Plants)

> summary(mod.fit)

Call:

glm(formula = Virus8/Plants ~ Infest + Control, family = binomial(link = logit), data = tomato, weights = Plants)

Deviance Residuals:

 Min 1Q Median 3Q Max

-4.288 -2.425 -1.467 1.828 8.379

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) -0.6652 0.1018 -6.533 6.45e-11 \*\*\*

Infest2 0.2196 0.1091 2.013 0.0441 \*

ControlC -0.7933 0.1319 -6.014 1.81e-09 \*\*\*

ControlN 0.5152 0.1313 3.923 8.74e-05 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 278.69 on 15 degrees of freedom

Residual deviance: 183.27 on 12 degrees of freedom

AIC: 266.77

Number of Fisher Scoring iterations: 4

Because the response variable is given in a binomial form, we used the weight argument along with the success/trials formulation in the formula argument. The estimated logistic regression models is



where we use the informal notation of “Infest2” to represent the indicator variable of level 2 for Infest and “C” and “N” to represent the indicator variables for levels C and N of Control, respectively. One could use “ControlC” and “ControlN” for “C” and “N”, respectively, if it were less obvious what these indicator variables represent.

Based on the positive estimated parameter for Infest2, the probability of showing symptoms is estimated to be larger in greenhouses where infestation method #2 is used. Also, based on the estimated parameters for C and N, the estimated probability of showing symptoms is lowest for the chemical control method and highest for when no control method is used. These interpretations rely on their not being an interaction between the explanatory variables in the model. We next examine how to include the possibility of these interactions along with evaluating their importance.

Hypothesis tests for categorical explanatory variables

As with normal linear regression, all indicator variables must be included in a hypothesis test to evaluate the importance of a categorical explanatory variable.

Consider again the example with the categorical explanatory variable having 4 levels:



To evaluate the importance of this explanatory variable, we need to test

H0: β1 = β2 = β3 = 0

Ha: At least one β not equal to 0

Three separate Wald tests of H0: βi = 0 vs. Ha: βi ≠ 0 are not appropriate. One could do one overall Wald test, but we will focus instead on using a LRT because it performs better.

Interactions including categorical explanatory variables

Multiply each indicator variable by the model terms representing the other explanatory variable(s)

Questions:

* How would you represent an interaction between a categorical explanatory variable with 4 levels and one continuous explanatory variable?
* How would you perform a test to evaluate the interaction in the previous example question?
* How would you represent an interaction between a categorical explanatory variable with 4 levels and another categorical explanatory variable with 3 levels?

Example: Control of the Tomato Spotted Wilt Virus (TomatoVirus.R, TomatoVirus.csv)

In actual application, it is important to consider the possibility of an interaction between the infestation method and the control method. The code below shows how to include this interaction and evaluate its importance:

> mod.fit.inter<-glm(formula = Virus8/Plants ~ Infest +

 Control + Infest:Control, family = binomial(link =

 logit), data = tomato, weight = Plants)

> summary(mod.fit.inter)

Call:

glm(formula = Virus8/Plants ~ Infest + Control + Infest:Control, family = binomial(link = logit), data = tomato, weights = Plants)

Deviance Residuals:

 Min 1Q Median 3Q Max

-3.466 -2.705 -1.267 2.811 6.791

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) -1.0460 0.1316 -7.947 1.92e-15 \*\*\*

Infest2 0.9258 0.1752 5.283 1.27e-07 \*\*\*

ControlC -0.1623 0.1901 -0.854 0.393

ControlN 1.1260 0.1933 5.826 5.68e-09 \*\*\*

Infest2:ControlC -1.2114 0.2679 -4.521 6.15e-06 \*\*\*

Infest2:ControlN -1.1662 0.2662 -4.381 1.18e-05 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 278.69 on 15 degrees of freedom

Residual deviance: 155.05 on 10 degrees of freedom

AIC: 242.55

Number of Fisher Scoring iterations: 4

> library(package = car)

> Anova(mod.fit.inter)

Analysis of Deviance Table (Type II tests)

Response: Virus8/Plants

 LR Chisq Df Pr(>Chisq)

Infest 4.060 1 0.0439 \*

Control 91.584 2 < 2.2e-16 \*\*\*

Infest:Control 28.224 2 7.434e-07 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> mod.fit$xlevels #Another way to see the levels of

 categorical explanatory variables

$Infest

[1] "1" "2"

$Control

[1] "B" "C" "N"

The estimated logistic regression model with the interaction is:



A LRT to evaluate the importance of the interaction term tests the parameters corresponding to Infest2×C and Infest2×N, say β4 and β5. The hypotheses are

H0: β4 = β5 = 0

Ha: At least one β not equal to 0

Notice that H0 is the model that still includes the main effect terms. The test statistic is -2log(Λ) = 28.224, and the p-value is 7.4×10-7 using a  approximation. Thus, there is strong evidence of an interaction between infestation method and control.

Question: What hypotheses do the p-value for Infest in the Anova() output correspond to?

Odds ratios

Odds ratios are useful for interpreting a categorical explanatory variable in a model; however, they are easily misinterpreted. For example, suppose we want to interpret  from the model



in the example with the categorical explanatory variable having 4 levels. A common **mistake** is to interpret this odds ratio as

The odds of a success are  times as large as for level B than *all of the other levels*

where the mistake is italicized. To see why this is wrong, note that the odds of a success at level B are



In order to have a resulting OR = , we need the denominator of the odds ratio to be . Thus, x1 = x2 = x3 = 0 for this second odds, which corresponds to level A of the categorical variable. The correct interpretation of the odds ratio is then

The odds of a success are  times as large as for level B than for level A

Similar interpretations are found for  (compare level C to A) and for  (compare level D to A).

What if you would like to compare level B to level C? You need to find the ratio of two odds:



Similarly, the odds ratios can be found for comparing level B to level D as  and level C to level D as .

Again, please remember that an odds ratio is just the ratio of two odds. Whenever you have difficulty understanding an odds ratio, go back to the basics and form the ratio.

Question: What would be the equation for the Wald confidence interval for the odds ratio comparing level B to level C?

Example: Control of the Tomato Spotted Wilt Virus (TomatoVirus.R, TomatoVirus.csv)

Because the interaction between infestation and control is significant, we would normally focus only on the model that includes the interaction. However, it is instructive first to see how calculations are performed using the model without the interaction. We will later perform the more complicated investigations with the model including the interaction.

Without interaction

The estimate model is



The estimated odds ratios for the control methods are:

> exp(mod.fit$coefficients[3:4])

ControlC ControlN

0.452342 1.674025

> #Control N vs. Control C

> exp(mod.fit$coefficients[4] - mod.fit$coefficients[3])

ControlN

3.700795

For example, the estimated odds ratio comparing level N to level B is e0.5152 = 1.67. Thus,

The estimated odds of plants showing symptoms are 1.67 times as large for using no control methods than using a biological control, where the infestation method is held constant

Because we would prefer to REDUCE the proportion of plants showing symptoms, it may be of more interest to invert the odds ratio:

The estimated odds of plants showing symptoms are 1/1.67 = 0.5973 times as large for using a biological control method than using no control methods, where the infestation method is held constant. Thus, using the spider mites (biological control) is estimated to reduce the odds of a plant showing symptoms by approximately 40%.

In order to compare the no control to the chemical control, the odds ratio is



The estimated odds ratio is e0.5152-(-0.7933) = 3.70.

Question: Which method – B or C – reduces the estimated odds more?

Below is the code to find the corresponding confidence intervals:

> # Using mcprofile

> library(package = mcprofile)

> K<-matrix(data = c(0, 0, 1, 0,

 0, 0, 0, 1), nrow = 2, ncol = 4,

 byrow = TRUE)

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

> ci.log.OR<-confint(object = linear.combo, level = 0.95,

 adjust = "none")

> ci.log.OR

 mcprofile - Confidence Intervals

level: 0.95

adjustment: none

 Estimate lower upper

C1 -0.793 -1.054 -0.536

C2 0.515 0.258 0.773

> comparison<-c("C vs. B", "N vs. B")

> data.frame(comparison, OR = exp(ci.log.OR$confint))

 comparison OR.lower OR.upper

1 C vs. B 0.3486325 0.5848772

2 N vs. B 1.2945688 2.1665987

> save.wald<-wald(object = linear.combo)

> ci.logit.wald<-confint(object = save.wald, level = 0.95,

 adjust = "none")

> data.frame(comparison, lower =

 exp(ci.logit.wald$confint[,1]), upper =

 exp(ci.logit.wald$confint[,2]))

 comparison lower upper

1 C vs. B 0.3492898 0.5857982

2 N vs. B 1.2941188 2.1654579

For example, the 95% profile LR confidence interval comparing level N to level B is 1.29 to 2.17. Thus,

With 95% confidence, the odds of plants showing symptoms are between 1.29 and 2.17 times as large when using no control methods rather than using a biological control (holding the infestation method constant).

Alternatively, we could also say

With 95% confidence, the odds of plants showing symptoms are between 0.46 and 0.77 times as large when using a biological control method rather than using no control methods (holding the infestation method constant). Thus, using the spider mites (biological control) is estimated to reduce the odds of a plant showing symptoms by approximately 23% to 54%.

Without using mcprofile, we could find these intervals as follows:

> #Profile likelihood ratio interval

> exp(confint(object = mod.fit, parm = c("ControlC",

 "ControlN"), level = 0.95))

Waiting for profiling to be done...

 2.5 % 97.5 %

ControlC 0.3486313 0.5848759

ControlN 1.2945744 2.1666574

> #Wald interval

> exp(confint.default(object = mod.fit, parm =

 c("ControlC", "ControlN"), level = 0.95))

 2.5 % 97.5 %

ControlC 0.3492898 0.5857982

ControlN 1.2941188 2.1654579

In order to compare no control to chemical control, we need to use a Wald interval of



where



Below is the code for mcprofile involving a profile LR and a Wald interval:

> K<-matrix(data = c(0, 0, -1, 1), nrow = 1, ncol = 4,

 byrow = TRUE)

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

> ci.log.OR<-confint(object = linear.combo, level = 0.95,

 adjust = "none")

> ci.log.OR

 mcprofile - Confidence Intervals

level: 0.95

adjustment: none

 Estimate lower upper

C1 1.31 1.03 1.59

> data.frame(comparison = "N vs. C", OR =

 exp(ci.log.OR$confint))

 comparison OR.lower OR.upper

C1 N vs. C 2.805262 4.900458

> save.wald<-wald(object = linear.combo)

> ci.logit.wald<-confint(object = save.wald, level =

 0.95, adjust = "none")

> data.frame(lower = exp(ci.logit.wald$confint[,1]),

 upper = exp(ci.logit.wald$confint[,2]))

 lower upper

1 2.800422 4.890649

In practice, I would have included a row of 0, 0, -1, 1 in the first K matrix (making it a 3×4 matrix). I just decided to separate out this row here because it was a little different than the other two rows.

Without using mcprofile, we could find these intervals as follows:

Method #1: Program in formula for Wald.

> #Wald interval for Control N vs. Control C

> beta.hat<-mod.fit$coefficients[-1] #Matches up beta

 indices with [i] to help avoid mistakes

> exp(beta.hat[3] - beta.hat[2])

ControlN

3.700795

> cov.mat<-vcov(mod.fit)[2:4,2:4]

> var.N.C<-cov.mat[3,3] + cov.mat[2,2] - 2\*cov.mat[3,2]

> CI.betas<- beta.hat[3]- beta.hat[2] + qnorm(p = c(0.025,

 0.975))\*sqrt(var.N.C)

> exp(CI.betas)

[1] 2.800422 4.890649

Method #2: Change the base level to C.

> tomato$Control.reorder<-relevel(x = tomato$Control, ref =

 "C")

> mod.fit2<-glm(formula = Virus8/Plants ~ Infest +

 Control.reorder, family = binomial(link = logit), data

 = tomato, weight = Plants)

> summary(mod.fit2)

Call:

glm(formula = Virus8/Plants ~ Infest + Control.reorder, family = binomial(link = logit), data = tomato, weights = Plants)

Deviance Residuals:

 Min 1Q Median 3Q Max

-4.288 -2.425 -1.467 1.828 8.379

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) -1.4585 0.1164 -12.526 < 2e-16 \*\*\*

Infest2 0.2196 0.1091 2.013 0.0441 \*

Control.reorderB 0.7933 0.1319 6.014 1.81e-09 \*\*\*

Control.reorderN 1.3085 0.1422 9.200 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 278.69 on 15 degrees of freedom

Residual deviance: 183.27 on 12 degrees of freedom

AIC: 266.77

Number of Fisher Scoring iterations: 4

> #Wald interval

> exp(confint.default(object = mod.fit2, parm =

 c("Control.reorderB", "Control.reorderN"), level =

 0.95))

 2.5 % 97.5 %

Control.reorderB 1.707073 2.862952

Control.reorderN 2.800422 4.890649

> c(1/2.8629521, 1/1.707073)

[1] 0.3492898 0.5857980

> #Profile likelihood ratio interval

> exp(confint(object = mod.fit2, parm =

 c("Control.reorderB", "Control.reorderN"), level =

 0.95))

Waiting for profiling to be done...

 2.5 % 97.5 %

Control.reorderB 1.709764 2.868360

Control.reorderN 2.805268 4.900565

Again, profile LR intervals are preferred over Wald intervals, but we can see they are very similar due to the large sample sizes.

Odds ratios can also be calculated for the Infest variable too. These calculations are given in the corresponding program to this example.

With interaction

The estimated model is



To understand the effect of Control on the response, we will need to calculate odds ratios where the level of Infest2 is fixed at either 0 or 1. The odds ratio comparing level N to level B with Infest2 = 0 is



The odds ratio comparing level N to level B with Infest2 = 1 is



Other odds ratios can be calculated in a similar manner. Below are all of the estimated odds ratios and corresponding confidence intervals for Control holding Infest2 constant:

> K<-matrix(data = c(0, 0, 0, 1, 0, 0,

 0, 0, 0, 1, 0, 1,

 0, 0, 1, 0, 0, 0,

 0, 0, 1, 0, 1, 0,

 0, 0, -1, 1, 0, 0,

 0, 0, -1, 1, -1, 1), nrow = 6, ncol

 = 6, byrow = TRUE)

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

> ci.log.OR<-confint(object = linear.combo, level = 0.95,

 adjust = "none")

> ci.log.OR

 mcprofile - Confidence Intervals

level: 0.95

adjustment: none

 Estimate lower upper

C1 1.1260 0.750 1.508

C2 -0.0402 -0.400 0.318

C3 -0.1623 -0.536 0.210

C4 -1.3738 -1.750 -1.009

C5 1.2884 0.905 1.678

C6 1.3336 0.934 1.742

> data.frame(Infest2 = c(0, 1, 0, 1, 0, 1), comparison, OR

 = round(exp(ci.log.OR$estimate),2), OR.CI =

 round(exp(ci.log.OR$confint),2))

 Infest2 comparison Estimate OR.CI.lower OR.CI.upper

C1 0 N vs. B 3.08 2.12 4.52

C2 1 N vs. B 0.96 0.67 1.37

C3 0 C vs. B 0.85 0.58 1.23

C4 1 C vs. B 0.25 0.17 0.36

C5 0 N vs. C 3.63 2.47 5.36

C6 1 N vs. C 3.79 2.54 5.71

> save.wald<-wald(object = linear.combo)

> ci.logit.wald<-confint(object = save.wald, level = 0.95,

 adjust = "none")

> data.frame(Infest2 = c(0, 1, 0, 1, 0, 1), comparison, OR

 = round(exp(ci.log.OR$estimate),2), lower =

 round(exp(ci.logit.wald$confint[,1]),2), upper =

 round(exp(ci.logit.wald$confint[,2]),2))

 Infest2 comparison Estimate lower upper

C1 0 N vs. B 3.08 2.11 4.50

C2 1 N vs. B 0.96 0.67 1.38

C3 0 C vs. B 0.85 0.59 1.23

C4 1 C vs. B 0.25 0.17 0.37

C5 0 N vs. C 3.63 2.46 5.34

C6 1 N vs. C 3.79 2.53 5.68

The columns of K are ordered corresponding to the 6 parameters estimated by the model. For example, row 2 corresponds to estimating



where the 4th and 6th columns of K have 1’s for the 4th and 6th parameters. Remember the first parameter in the model is β0 so this is why the column numbers are 1 higher than the indices for the β’s.

The estimated odds ratio comparing level N to level B with Infest2 = 1 is e1.1260 – 1.1662 = 0.96. Thus,

The estimated odds of plants showing symptoms are 0.96 times as large for using no control than using a biological control when infected thrips are released into the greenhouse.

We can also see why the interaction between Infest and Control was significant. The N vs. B and C vs. B odds ratio differ by a large amount over the two levels of Infest. However, there is not much of a difference for N vs. C over the levels of Infest.

The 95% profile likelihood ratio interval comparing level N to level B with Infest2 = 1 is 0.67 < OR < 1.38. Thus,

With 95% confidence, the odds of plants showing symptoms are between 0.67 and 1.38 times as large for using no control methods than using a biological control when infected thrips are released in the greenhouse. Because 1 is within the interval, there is not sufficient evidence to conclude a biological control is effective in this setting.

Notice the interval for comparing level N to level B with Infest2 = 0 is 2.11 < OR < 4.50. Because the interval is above 1, there is sufficient evidence to conclude the biological control reduces the odds of plants showing symptoms when interspersing infected plants with uninfected thrips.

Comments:

* MANY more interpretations would be needed in a real-life setting. For example, one should interpret the infest variable for fixed levels of the control variable.
* My book shows how to control the familywise error rate.

**Section 2.2.7 – Convergence of parameter estimates**

The glm() function uses IRLS until convergence is obtained or until the maximum number of iterations are reached. To determine convergence, glm() does not look at the successive estimates of the parameters directly, rather it examines the residual deviance. If we let G(k) denote the residual deviance at iteration k, then convergence occurs when



where ε is some specified small number greater than 0. The numerator provides a measure to determine if the  for i = 1, …, n are changing much from one iteration to the next. The denominator helps to take into account the relative size of the residual deviance.

The glm() function provides a few ways to control how convergence decided:

* The epsilon argument sets ε above. The default is epsilon = 10^(-8).
* The maxit argument states the maximum number of iterations allowed for the numerical procedure. The default is maxit = 25.
* The trace argument value can be used to see the actual G(k) values for each iteration. The default is trace = FALSE (do not show these values).

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

Consider the model with only distance as the explanatory variable:



Below are the results from using the glm() to estimate the model and from including the three arguments controlling convergence. Note that these three argument values were chosen for illustrative purposes only:

mod.fit<-glm(formula = good ~ distance, family =

 binomial(link = logit), data = placekick, trace = TRUE,

 epsilon = 0.0001, maxit = 50)

Deviance = 836.7715 Iterations - 1

Deviance = 781.1072 Iterations - 2

Deviance = 775.8357 Iterations - 3

Deviance = 775.7451 Iterations - 4

Deviance = 775.745 Iterations - 5

> mod.fit$control

$epsilon

[1] 1e-04

$maxit

[1] 50

$trace

[1] TRUE

> mod.fit$converged

[1] TRUE

The convergence criteria value for iteration k = 5 is



which is less than the stated ε = 0.0001, so the iterative numerical procedure stopped. For iteration k = 4, the convergence criteria value is 0.00012, which is greater than 0.0001, so this is why the procedure continued.

If the value for maxit was changed to 3, the message

Warning message:

glm.fit: algorithm did not converge

would be printed to warn you that convergence was not obtained. Of course, you would NOT use the parameter estimates in this situation!

What if convergence is not obtained?

Try a larger number of iterations!

If this does not work, there may be some fundamental problems with the data making the iterative numerical procedures not suitable. The most common problem occurs when an explanatory variable(s) perfectly separates the data between y = 0 and 1 values; this is often referred to as *complete separation*. In addition to a convergence warning message, another warning message that glm() will likely give for this situation is

glm.fit: fitted probabilities numerically 0 or 1 occurred

This message corresponds to  = 0 or 1 for some observations. Please see the plot in the next example if it is not clear why this happens. Note that this message can be given in situations where there is not a problem too.

Example: Complete separation (Non-convergence.R)

Consider a simple data set with one explanatory variable x1 that is less than 6 when y = 0 and greater than or equal to 6 when y = 1. Because x1 perfectly separates out the two possible values of y, complete separation occurs. Below is the corresponding R code and output:

> set1<-data.frame(x1 = c(1,2,3,4,5,6,7,8,9,10), y =

 c(0,0,0,0,0,1,1,1,1,1))

> set1

 x1 y

1 1 0

2 2 0

<OUTPUT IS EDITED>

10 10 1

> mod.fit1<-glm(formula = y ~ x1, data = set1, family = binomial(link = logit), trace = TRUE)

Deviance = 4.270292 Iterations - 1

Deviance = 2.574098 Iterations - 2

<OUTPUT IS EDITED>

Deviance = 7.864767e-10 Iterations - 25

Warning messages:

1: glm.fit: algorithm did not converge

2: glm.fit: fitted probabilities numerically 0 or 1 occurred

> mod.fit1$coefficients

(Intercept) x1

 -245.84732 44.69951

R indicates that both convergence did not occur and at least some estimates of π are 0 or 1. Below is a plot (left side) of the data and the model at iteration #25:



Because there is a separation between the y = 0 and 1 values, the slope of the line between x = 5 and 6 will continue to get larger as the iterations continue. Essentially, the β1 estimate is going to infinity with continued iterations. Notice this means the estimate is VERY biased.

Interestingly, R indicates “convergence” after 26 iterations if you increase maxit! However, the same

glm.fit: fitted probabilities numerically 0 or 1 occurred

message will occur. You should not use logistic regression here because the parameter estimates will continue to change for a larger number of iterations. Try this yourself with a larger maxit and smaller epsilon.

By reversing the y values at x1 = 5 and 6, we obtain model convergence in 6 iterations (not shown here). The right plot above shows the data and the final model. The slope of the model is now not as great as was before.

What can you do if complete separation occurs?

Heinze and Schemper (2002) outline a number of possible options. The most desirable options are

* Exact logistic regression – The exact distribution of the β estimators is used through the use of computational intensive algorithms. Chapter 6 of my book discusses exact inference methods.
* Modify the likelihood function – Because the likelihood function increases without bound during the iterative numerical procedure, this function can be modified to potentially prevent the problems from happening. One approach is to include a “penalty” in the likelihood function as proposed by Firth (1993). Details are given in the corresponding section of my book.

Final comments:

* Complete separation is not necessarily bad if you want to distinguish between the response levels of y. The problem is that the model estimated by maximum likelihood does not provide a good way to interpret the relationship between y and the explanatory variables.
* It can be difficult to see complete separation graphically if there is more than one explanatory variable. There may be times even when the glm() function does not provide a warning. When parameter estimates are very large or very small with large estimated standard deviations, this is sign that complete separation may exist. These types of parameter estimates can then lead to many observations with estimated probabilities of success close to 0 or 1.
* An often used solution to complete separation is to add a pseudo observation to the data in an attempt to obtain convergence of the parameter estimates. This is somewhat similar to what was done for odds ratios with a 0 cell count present within a contingency table. We recommend using the exact logistic regression or penalized likelihood approaches instead.
* The glm2 package uses a different model fitting method. See the December 2011 issue of *The R Journal* for more information. Note that this function will not necessarily solve complete separation problems.

**Section 2.2.8 – Monte Carlo simulation**

This section examines how to use Monte Carlo simulation to evaluate inference procedures in a similar manner as we saw in Chapter 1. This section also helps to give intuition for why we can use particular probability distributions to help perform Wald and likelihood ratio based inferences.

**Section 2.3 – Generalized linear models**

Logistic regression models fall within a family of models called *generalized linear models*. Each generalized linear model has three different components:

1. Random: This specifies the distribution for Y. For the logistic regression model, Y has a Bernoulli distribution.
2. Systematic: This specifies a linear combination of the β parameters with the explanatory variables, and it is often referred to as the linear predictor. For the logistic regression model, we have β0 + β1x1 + βpxp.
3. Link: This specifies how the expected value of the random component E(Y) is linked to the systematic component. For the logistic regression model, we have  where E(Y) = π and the logit transformation is the link function.

Note that “linear” in generalized linear models comes from the β parameters simply being coefficients for the explanatory variables in the model. Nonlinear models involve more complex functional forms such as xβ.

Link functions used with binary regression models

There are many other models than logistic regression that belong to this family. With respect to models for binary responses, they have the same random and systematic components as a logistic regression model. The link function is what is different, where the most important aspect is to have an *inverse* link function that guarantees 0 < π < 1. For example, the inverse link function for logistic regression is the transformation:



Inverse link functions used in practice are based on cumulative distribution functions (CDFs) for random variables. The CDF of a logistic probability distribution is used for logistic regression, which results in the name for the model.

CDF Review:

Let X be a continuous random variable with probability density function f(x). An observed value of X is denoted by x. The CDF of X is F(x) = P(Xx) = . Note that u is substituted into the probability density function to avoid confusion with the upper limit of integration. If X is a discrete random variable, the CDF of X is F(x) = P(Xx) =  =  where the sum is over all values of X ≤ x.

An informal definition is the CDF “cumulates” probabilities as a function of x.

The reason why CDFs are used as link functions for binary data is because the CDF is always between 0 and 1.

We have used CDFs many times in R already through the use of functions such as pbinom(), pnorm(), and pchisq(). For example, pnorm(q = 1.96) is expressed as F(1.96) where F(⋅) is the CDF of a standard normal distribution for this case. Equivalently, Z0.975 = 1.96.

Example: Logistic probability distribution (Logistic.R)

A random variable X with a logistic probability distribution has a probability density function of



for -∞ < x < ∞ and parameters -∞ < μ < ∞ and σ > 0. The CDF is



Below is a plot of the PDF and CDF for μ = -2 and σ = 2.



> mu<--2

> sigma<-2

> #Examples for f(-2) and F(-2)

> dlogis(x = -2, location = mu, scale = sigma)

[1] 0.125

> plogis(q = -2, location = mu, scale = sigma)

[1] 0.5

> x11(width = 10, height = 6, pointsize = 12) #win.graph()

 could be used too

> par(mfrow = c(1,2))

> par(pty="s")

> curve(expr = dlogis(x = x, location = mu, scale = sigma),

 ylab = "f(x)", xlab = "x", xlim = c(-15, 15), main =

 "PDF", col = "black",, n = 1000)

> abline(h = 0)

> curve(expr = plogis(q = x, location = mu, scale = sigma),

 ylab = "F(x)", xlab = "x", xlim = c(-15, 15), main =

 "CDF", col = "black", n = 1000)

We see that the distribution is symmetric and centered at μ. Larger values of σ lead to a larger variance. Overall, E(X) = μ and Var(X) = σ2π2/3. In comparison to a normal distribution with mean μ and variance σ2, the distributions are centered at the same location, but the logistic distribution has a larger variance.

The CDF plot above is actually the same plot as on page 7 in our initial example of a logistic regression model! Notice that



when β0 = -μ/σ and β1 = 1/σ. With μ = -2 and σ = 2, we obtain β0 = 1 and β1 = 0.5.

If we solve for β0 + β1x in the above equation, we obtain



Another way to write the left side of this equation is to use F-1(x), which is the inverse CDF of X. Therefore, the link function of a logistic regression model is the inverse CDF of the logistic probability distribution.

While the logit link function is the most prevalently used for binary regression, there are two other functions that are common:

* Inverse CDF of a standard normal distribution: This produces what is known as a probit regression model.

Suppose Φ(⋅) denotes the CDF of a standard normal distribution. The model is written as

π = Φ(β0 + β1x1 + βpxp)

or equivalently as

Φ-1(π) = β0 + β1x1 + βpxp

A very common way to express the model is

probit(π) = β0 + β1x1 + βpxp

where probit is used to denote the inverse CDF transformation in a similar manner as the logit transformation is for logistic regression. The term probit is a shortened version of “probability unit” (Hubert, 1992). Chapter 8 of Salsburg (2001) gives an interesting account of how this model was developed by Chester Bliss.

* Inverse CDF of a Gumbel (extreme value) distribution: This produces what is known as a complementary log-log regression model.

The CDF is



where -∞ < x < ∞ and parameters -∞ < μ < ∞ and σ > 0. Note that 1 – F(x) is also between 0 and 1. Using this result, the model is

π = 1 – exp[-exp(β0 + β1x1 + βpxp)]

Solving for the linear predictor, we obtain

log[-log(1 – π)] = β0 + β1x1 + βpxp

Thus, the model's name comes from using 1 – π

 (which is like using P(Ac) = 1-P(A) where Ac is the complement of an event A) and a double log transformation.

Example: Compare three binary regression models (PiPlot.R)

Suppose the linear predictor has only one explanatory variable of the form β0 + β1x. By choosing values of β0 and β1 so that the mean is 2 and the variance is 3 for the corresponding CDFs, we obtain the plots of the models displayed below (see the corresponding program for code):



Comments:

* The logistic and probit regression models are symmetric about π = 0.5; the complementary log-log model is not symmetric.
* The logistic and probit models are quite similar.
* The logistic model rises a little more slowly toward π = 1 and falls a little more slowly to π = 0. This characteristic is due to a logistic probability distribution having more probability in its tails than a normal distribution when both are centered at the same location (i.e., larger variance).

Estimation and inference for binary regression models

Estimation: Probit and complementary log-log models are estimated in the same way as the logistic regression model. The difference now is that π is represented in the log-likelihood function by the corresponding probit or complementary log-log model specification.

Inference: Once the parameter estimates are found, the same inference procedures as used for the logistic regression model are available for the probit and complementary log-log models.

Odds ratios: Odds ratios are not as easy to interpret with probit and complementary log-log regression models as they were for logistic regression models. For a logistic regression model of logit(π) = β0 + β1x, we saw earlier than the odds ratio was simply



for ANY c-unit increase in x. The same simplification does not hold true for probit and complementary log-log models. This is one of the main reasons why logistic regression models are the most used binary regression models.

Example: Odds ratios used with probit models

Consider the model probit(π) = β0 + β1x. The odds of a success are



at a particular value of x. The odds of a success with a c-unit increase in x are



When the odds ratio is formed from these two odds, x will remain in the final expression! Therefore, the odds ratio for a probit regression model depends on the value of the explanatory variable.

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

Using distance as the only explanatory variable, below is how you can estimate the probit and complementary log-log models in R:

> #Logistic

> mod.fit.logit<-glm(formula = good ~ distance, family =

 binomial(link = logit), data = placekick)

> summary(mod.fit.logit)

Call:

glm(formula = good ~ distance, family = binomial(link = logit), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

-2.7441 0.2425 0.2425 0.3801 1.6092

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 5.812080 0.326277 17.81 <2e-16 \*\*\*

distance -0.115027 0.008339 -13.79 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.43 on 1424 degrees of freedom

Residual deviance: 775.75 on 1423 degrees of freedom

AIC: 779.75

Number of Fisher Scoring iterations: 6

> #Probit

> mod.fit.probit<-glm(formula = good ~ distance, family =

 binomial(link = probit), data = placekick)

> summary(mod.fit.probit)

Call:

glm(formula = good ~ distance, family = binomial(link = probit), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

-2.8171 0.2274 0.2274 0.3913 1.5320

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 3.207118 0.157043 20.42 <2e-16 \*\*\*

distance -0.062796 0.004318 -14.54 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.43 on 1424 degrees of freedom

Residual deviance: 771.95 on 1423 degrees of freedom

AIC: 775.95

Number of Fisher Scoring iterations: 6

> #Complementary log-log

> mod.fit.cloglog<-glm(formula = good ~ distance, family =

 binomial(link = cloglog), data = placekick)

> summary(mod.fit.cloglog)

Call:

glm(formula = good ~ distance, family = binomial(link = cloglog), data = placekick)

Deviance Residuals:

 Min 1Q Median 3Q Max

-2.9054 0.2125 0.2125 0.4132 1.3706

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) 2.380155 0.118409 20.10 <2e-16 \*\*\*

distance -0.052232 0.003711 -14.08 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1013.43 on 1424 degrees of freedom

Residual deviance: 769.48 on 1423 degrees of freedom

AIC: 773.48

Number of Fisher Scoring iterations: 5

The estimated models are

* 
* 
* 

You may think there are large differences among the models due to their parameter estimates. However, examining the parameter estimates alone does not take into account the functional form of the models. Below are some comparisons of their estimated probabilities of success

> #Compare pi^ values without predict()

> distance<-c(20, 35, 50)

> #Logistic

> plogis(q = mod.fit.logit$coefficients[1] +

 mod.fit.logit$coefficients[2] \* distance)

[1] 0.9710145 0.8564542 0.5151820

> #Probit

> pnorm(q = mod.fit.probit$coefficients[1] +

 mod.fit.probit$coefficients[2] \* distance)

[1] 0.9744831 0.8435734 0.5268331

> #Complementary log-log

> 1-exp(-exp(mod.fit.cloglog$coefficients[1] +

 mod.fit.cloglog$coefficients[2] \* distance))

[1] 0.9776725 0.8239115 0.5476850

> #Compare pi^ values with predict() (easier)

> predict.data<-data.frame(distance = c(20, 35, 50))

> logistic.pi<-predict(object = mod.fit.logit, newdata =

 predict.data, type = "response")

> probit.pi<-predict(object = mod.fit.probit, newdata =

 predict.data, type = "response")

> cloglog.pi<-predict(object = mod.fit.cloglog, newdata =

 predict.data, type = "response")

> round(data.frame(predict.data, logistic.pi, probit.pi,

 cloglog.pi),4)

 distance logistic.pi probit.pi cloglog.pi

1 20 0.9710 0.9745 0.9777

2 35 0.8565 0.8436 0.8239

3 50 0.5152 0.5268 0.5477

The estimates are quite similar. We can further investigate this through plotting the models (see program for code):



We see the models are quite similar except for at larger distances when comparing the complementary log-log model to the logistic and probit models.

Please see the odds ratios calculations shown in the book. These calculations demonstrate that the odds ratios change as a function of distance.

Random components used with generalized linear models

We have only discussed using a Bernoulli distribution for Y as part of the random component. There are many other probability distributions that can be used depending on the response type. For example, the normal linear regression model of

E(Y) = β0 + β1x1 + β2x2 + … + βpxp

uses the normal distribution for Y along with an identity link function (E(Y) equals the linear predictor). Also, in Chapter 4, we will examine count responses, such as 0, 1, 2, … , without any per-determined limits. In this setting, a model frequently used is

log[E(Y)] = β0 + β1x1 + β2x2 + … + βpxp

where Y has a Poisson distribution and a log link function is used. This log link function ensures that E(Y) > 0 for any combination of parameters and explanatory variable values.