**Chapter 6 – Additional topics**

**Section 6.5 – Mixed models and estimating equations for correlated data**

All of the previous chapters assumed each observation was independent. In some situations, this assumption may not be true. Below are some examples where observations are “grouped” in some manner that leads likely to dependence:

* Pigs in a pen – A classic example from an experimental design course will discuss pigs in a pen where multiple pens are available. Responses from pigs in the same pen are likely correlated. A “block” could be used to account for the pen depending on the setting.
* Longitudinal data – Data from the same individual may be collected at time points 0, 1, 2, …, and multiple individuals contribute observations to the same data set. The responses are repeatedly observed on the same individual so they are likely correlated.
* Choose all that apply data – Survey questions often ask respondents to “Choose all responses that apply from the following set of items.” Suppose there are c possible items to choose from. Whether a particular response is chosen or not represents a binary outcome. The c binary outcomes are likely correlated for the same individual.
* Placekicking data – Each placekicker contributes multiple observations. The responses obtained from the same individual could be correlated.

Responses observed in clusters (e.g., pigs in a pen) or observed repeatedly on the same unit (e.g., longitudinal data) are likely more similar to each other than would be expected under independence. This usually leads to a positive correlation among the responses within the sampling unit. When this occurs and the data are treated as independent, this usually leads to liberal inferences. Also, this positive correlation leads to evidence of overdispersion.

**Sections 6.5.1 and 6.5.2 – Random effects and Mixed-effects models**

To help explain “random” and “fixed” effects, consider all possible individuals who could participate in a longitudinal data study for times 0, 1, …, t where the response measured at each time point is heart rate.

We can imagine that each individual has a constant value associated with him/her that gets added to the overall mean or probability, raising or lowering it by a unique amount. This constant value is called the random effect. Because there are a fixed number of individuals in a study, the set of all individual random effects here are referred to as a random-effects factor.

For the purposes of modeling, we typically assume that the random effects follow some known distribution. This distribution then allows us to conceptualize the individuals in the sample as being obtained from a larger population from which we would like to make inferences to.

Most commonly, we assume the random effects as being normally distributed with mean 0 and unknown variance. The zero mean ensures that the “average individual” adds 0 to the measurements taken on units within it, while the unknown variance creates a new parameter in the model, representing the amount by which the responses from different individuals may differ from one another. This variance is called a variance component.

Suppose for this longitudinal study that we also have a categorical explanatory variable called “drug” which is binary indicating whether a new drug or placebo was administered to an individual at the beginning of the study. This type of variable is often called a fixed-effects factor because the possible levels of it are either chosen deliberately or observed among a sample. Comparisons of means or probabilities among those specific levels are of interest. In particular, there is not assumed to be some larger population of levels from which the observed levels were chosen.

Mixed-effects models combine random and fixed effects into one regression model. We will discuss a specific type of these models known as a generalized linear mixed model (GLMM). Below is a simple GLMM for the longitudinal data study:

 (Model 1)

where

* Yik is the heart rate at time k for individual i; E(Yik) = μik
* β0 is an intercept parameter
* β1 is a slope parameter
* xi = 1 if individual i received the new drug at the beginning of the study; xi = 0 otherwise
* b0i is the random effect of individual i; b0i ~ independent N(0, )

To help emphasize the model structure, suppose we focus on individual #1. The responses for this individual would be modeled as follows:

 for time = 1

 for time = 2

 for time = t

The b01 term corresponds to the observed value of a random variable B01 with a N(0, ) distribution.

The responses for individual #2 would be modeled as follows:

 for time = 1

 for time = 2

 for time = t

The b02 term corresponds to the observed value of a random variable B02 with a N(0, ) distribution.

In the above models, notice what remains the same for each individual. Also, notice what changes across individuals.

Below are more general models:

*  – This allows for the drug application to change at each time point.
*  where B1i ~ independent N(0, ) (Model 2) – The random effect b1i term measures the amount by which the slope of xik in the linear predictor for individual i differs from the average slope β1.
* A bivariate normal distribution could be used in the last model for b0i and b1i which allows for dependence between the random variables. For example, individuals with small intercepts could tend to have small slopes leading to a positive correlation.
*  where C01, …, C0t have a multivariate normal distribution; this would be helpful to include time dependence.
* These types of models can be generalized in many other ways! Take the STAT 971 course for many more examples.

Example: Placekicking (PlacekickWithKickerGLMM.R, PlacekickWithNames.csv)

The data file is the same as we saw in Chapter 2, but now the placekicker’s name is included:

> kick <- read.csv("C:\\Data\\PlacekickWithNames.csv)

> head(kick)

 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

 date kicker

1 90395 BAHR

2 90395 BAHR

3 90395 STOVER

4 90395 BAHR

5 90395 COFER

6 90395 PELFREY

> table(kick$kicker)

ANDERSEN ANDERSON BAHR BIASUCCI BLANCHAR BONIOL

 50 42 46 23 35 54

 BRIEN BUTLER CARNEY CHRISTIE COFER DALUISO

 44 59 36 60 12 38

DELGRECO ELAM ELLIOTT G\_DAVIS HANSON HENTRICH

 58 52 44 39 60 9

 HOLLIS HUSTED JACKE KASAY LOHMILLE LOWERY

 48 40 45 44 17 32

MCLAUGLI MURRAY N\_JOHNSN PELFREY PETERSON REVEIZ

 22 48 54 56 47 62

 STOVER STOYANOV WILKINS ZENDEJAS

 42 62 33 12

Because each placekicker contributes multiple observations to the data set, it may be of interest to account for the potential dependence among each placekicker’s own placekicks. I did not when I originally analyzed the data during graduate school because:

1. Methods to take into account dependence were not as well developed for non-normally distributed responses!
2. Previous research had shown that placekickers were essentially treated by the NFL as “interchangeable parts” (Morrison and Kalwani, 1993). This made me much less concerned that there was any significant amount of dependence.

Consider the setting again where distance is the only explanatory variable. The model that takes into account a random effect for the placekicker is



where

* Yik = 1 for a success and 0 for a failure for placekick k of placekicker i; E(Yik) = πik
* β0 is an intercept parameter
* β1 is a slope parameter
* xik is the distance for placekick k of placekicker i
* b0i is the random effect of placekicker i; B0i ~ independent N(0, )

Note that this model formulation says that once the placekicker is taken into account, the responses for a placekicker are independent.

**Sections 6.5.3 – Model fitting**

GLMM’s are estimated through using likelihood methods. Unfortunately, the presence of random effects in the model make estimation much more difficult. This is because the true value of the random effect is unobservable!

Conditional on any random effects, the distribution of the response has the same form as we saw with generalized linear models. For instance, in the heart rate case, Yik has a Poisson distribution with mean μik but now notice the random effect plays a role. The problem then is that this μik only applies then to the ith person in the sample when we would ideally like to generalize to all people in the population!

The way around this problem is to find the maximum likelihood estimates based on the marginal distribution of the response variable where the random effects are “averaged over”. This is done by integrating out the random effects from a joint probability distribution.

For example, consider the heart rate case again with the model is



For individual i at time k, we have the following probability distribution for Yik:



where . This is the conditional distribution of Yik given B0i = b0i multiplied by the distribution of B0i. In a more mathematical statistics manner, this can be written as f(yik|b0i)×g(b0i) = h(yik,b0i).

For individual i, we can further write the joint distribution for all of their responses as



where we assume that the Yi1, …, Yit are independent once we know the random effect. In a more mathematical statistics manner, the above expression can be written as f(yi1, …, yit|b0i)×g(b0i) = h(yik, …, yit, b0i)

Of course, we don’t really know the random effect so let’s integrate it out:



Remember that μik is a function of b0i so this is not just an integral involving the normal distribution. Once the integration is completed, b0i is gone so we have just the distribution of Yi1, …, Yit. The likelihood function for all i = 1, …, n individuals is



This likelihood function is maximized as a function of the parameters to obtain maximum likelihood estimates!

The random effect terms can also be predicted. Notice this is different from estimating parameters (β0, β1, and σb0) because now we are interested in finding values for unobservable random variables. Through using joint and conditional distributions, we can find the distribution of b0i given the responses for the ith individual as



There are two ways this distribution could be used to determine a predicted value of b0i:

* Use  with the parameters replaced by estimates
* Use the mode of the distribution with the parameters replaced by estimates. The mode is simply the value of b0i that maximizes . This is what R does.

The integrals in the likelihood function and in  make these computations more difficult than we have seen in previous chapters. There are a few different approaches to handling the computations:

* + 1. Penalized quasi-likelihood or pseudo-likelihood (“approximate the model”): Using a Taylor-series approximation, find convenient approximations to the linear predictor that allow the model to be written as “mean + error,” the way a normal linear regression model is typically expressed. Using this approach, the integral that can be evaluated relatively easily using iterative numerical techniques.
		2. Laplace approximation (“approximate the integrand”): Because we assumed normal distributions for any random effects, the integrand has a form that can be approximated by a simpler function which is easier to integrate mathematically.
		3. Gaussian quadrature (“approximate the integral”): An integral in one dimension is an area under a curve. This area can be approximated by a sequence of rectangles, very much like approximating a density function with a histogram. Gaussian quadrature is a numerical method for performing this type of computation. The rectangles are referred to as the “points of quadrature.” Using a large number of quadrature points represents the shape of the integrand better than using fewer wide rectangles, but is also more computationally intensive.
		4. Bayesian: This puts estimation and inference into a different paradigm. Please see Section 6.6.

Page 427 of my book provides a nice discussion of the different methods. In summary:

* Gaussian quadrature is the most accurate when it computational can be done. Problems with actually being able to complete the calculations within a reasonable amount of time occur as the number of points of quadrature increase and number of random effects increase.
* Laplace approximations should be used when Gaussian quadrature is not possible. However, please note that these approximations are not as accurate. Laplace approximations are equivalent to Gaussian quadrature when there is only one point of quadrature.

We will focus on the use of the glmer() function in the lme4 package to perform maximum likelihood estimation (need to install yourself). It can use Gaussian quadrature for one random effect and Laplace approximations with one or more random effects.

The formula argument in glmer() allows one to specify the model like with glm(), but now the random effects needed to be included. If there is one explanatory variable x1 and one random effect grouping variable b in a data frame, the syntax for glmer() is

formula = y ~ *fixed effects* + (*fixed effects* | *random*

 *effects*)

where specific possible models include

* x1 + (1|b): β0 and β1 are estimated and a random effect is added to the intercept for each level of b (Model 1)
* x1 + (x1|b): β0 and β1 are estimated and random effects are added separately to the intercept and the regression coefficient for x1 at each level of b; these random effects are correlated (Could represent Model 2)
* x1 + (1|b) + (0 + x1|b): Same as above but now the two random effects are independent; if the 0 is removed, R will try to estimate the random effect for the intercept twice (Could represent Model 2)

Additional syntax help is given in the “Fitting linear mixed-effects models using lme4” vignette at <http://cran.r-project.org/web/packages/lme4/index.html>. In particular, the document shows that there are multiple ways to get the same models.

Example: Placekicking (PlacekickWithKickerGLMM.R, PlacekickWithNames.csv)

The purpose here is to estimate the model



Using 5 points of quadrature produces the following:

> library(lme4)

> mod.fit5 <- glmer(formula = good ~ distance +

 (1|kicker), nAGQ = 5, data = kick, family =

 binomial(link = "logit"))

> summary(mod.fit5)

Generalized linear mixed model fit by maximum likelihood ['glmerMod']

 Family: binomial ( logit )

Formula: good ~ distance + (1 | kicker)

 Data: kick

 AIC BIC logLik deviance

 781.0920 796.8778 -387.5460 775.0920

Random effects:

 Groups Name Variance Std.Dev.

 kicker (Intercept) 0.07633 0.2763

Number of obs: 1425, groups: kicker, 34

Fixed effects:

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

(Intercept) 5.875550 0.335020 17.54 <2e-16 \*\*\*

distance -0.116822 0.008461 -13.81 <2e-16 \*\*\*

---

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

Correlation of Fixed Effects:

 (Intr)

distance -0.948

The estimated model is



where bi0 is randomly sampled from N(0,0.0763).

To access components in mod.fit5, we can no longer use names(mod.fit5). The reason is due to glmer() using a S4-class structure rather than a S3-class structure that we have seen in the past.

R and its corresponding packages are written in a form very similar to the S programming language. Versions 3 and 4 of S are emulated by R. Version 3 (S3) is predominant. Version 4 is used by the glmer() function. My book discusses S3 vs. S4 with respect to the VGAM package in Chapter 3.

Instead of names(), we use slotNames():

> slotNames(mod.fit5)

 [1] "resp" "Gp" "call" "frame" "flist"

 [6] "cnms" "lower" "theta" "beta" "u"

[11] "devcomp" "pp" "optinfo"

> mod.glmm.5@beta

[1] 5.8755504 -0.1168217

Notice the use of @ rather than $ to access *slots* of the objects. Unfortunately, the estimate of the variance component () is not available within the mod.fit5 object, but we can access it using the usual S3 style with the summary() function:

> save.fit5 <- summary(mod.fit5)

> names(save.fit5)

 [1] "methTitle" "objClass" "devcomp"

 [4] "isLmer" "useScale" "logLik"

 [7] "family" "link" "ngrps"

[10] "coefficients" "sigma" "vcov"

[13] "varcor" "AICtab" "call"

[16] "residuals"

> save.fit5$varcor

 Groups Name Std.Dev.

 kicker (Intercept) 0.27627

> as.numeric(save.fit5$varcor)^0.5 #sigma^2 is hard to

 extract

[1] 0.2762724

What happens if we used a different number of quadrature points?

> quad.points <- c(1, 2, 4, 5, 10)

> save.estimates <- matrix(data = NA, nrow =

 length(quad.points), ncol = 3)

> counter <- 1

> for (quad in quad.points) {

 print(quad) #Turn off buffered output to see this prior

 to everything completing

 mod.fit.quad <- glmer(formula = good ~ distance +

 (1|kicker), nAGQ = quad, data = kick, family =

 binomial(link = "logit"))

 save.fit.quad <- summary(mod.fit5)

 save.estimates[counter,] <- round(c(mod.fit.quad@beta,

 as.numeric(save.fit.quad$varcor)^0.5),6)

 counter <- counter + 1

 }

[1] 1

[1] 2

[1] 4

[1] 5

[1] 10

> save.estimates <- as.data.frame(save.estimates)

> names(save.estimates) <- c("betahat0", "betahat1",

 "sigmahat2")

> data.frame(quad.points, save.estimates)

 quad.points betahat0 betahat1 sigmahat2

1 1 5.874802 -0.116801 0.276272

2 2 5.874886 -0.116802 0.276272

3 4 5.875557 -0.116822 0.276272

4 5 5.875550 -0.116822 0.276272

5 10 5.875550 -0.116822 0.276272

Notice that the Laplace approximation (1 quadrature point) does a good job with estimation.

Comments:

* The average value for the placekicker random effect is 0. Thus,  essentially gives us what we would obtain for the “average” placekicker at a particular distance. In Chapter 2, we obtain the following logistic regression model:



* The AIC for the logistic regression model was 779.7. The AIC for the logistic regression model with the random effect is 781.09. Which model is better by this information criterion?
* The functions ranef() and coef() provide information that remains the same for each kicker no matter the distance:

> head(ranef(mod.fit5)$kicker) #Estimates of bi0

 (Intercept)

ANDERSEN 0.186436602

ANDERSON -0.097883974

BAHR -0.101859113

BIASUCCI 0.003217527

BLANCHAR -0.002058115

BONIOL 0.108344955

> head(coef(mod.fit5)$kicker)

 (Intercept) distance

ANDERSEN 6.061987 -0.1168217

ANDERSON 5.777666 -0.1168217

BAHR 5.773691 -0.1168217

BIASUCCI 5.878768 -0.1168217

BLANCHAR 5.873492 -0.1168217

BONIOL 5.983895 -0.1168217

> mod.fit5@beta + head(ranef(mod.fit5)$kicker) #Same as

 above

 (Intercept)

ANDERSEN 6.061987021

ANDERSON -0.214705640

BAHR 5.773691307

BIASUCCI -0.113604138

BLANCHAR 5.873492305

BONIOL -0.008476711

* The predict() function is used to obtain the estimated probability of success:

> logit.i <- predict(object = mod.fit5, newdata = kick,

 REform = NULL, type = "link")

> logit.avg <- predict(object = mod.fit5, newdata =

 kick, REform = NA, type = "link")

> pi.hat.i <- predict(object = mod.fit5, newdata =

 kick, REform = NULL, type = "response")

> pi.hat.avg <- predict(object = mod.fit5, newdata =

 kick, REform = NA, type = "response")

> all <- data.frame(kicker = kick$kicker, distance =

 kick$distance, logit = round(logit.i, 3),

 logit.avg = round(logit.avg, 3), pi.hat.i =

 round(pi.hat.i, 3), pi.hat.avg = round(pi.hat.avg,

 3))

> head(all)

 kicker distance logit logit.avg pi.hat.i pi.hat.avg

1 BAHR 21 3.320 3.422 0.965 0.968

2 BAHR 21 3.320 3.422 0.965 0.968

3 STOVER 20 3.661 3.539 0.975 0.972

4 BAHR 28 2.503 2.605 0.924 0.931

5 COFER 20 3.430 3.539 0.969 0.972

6 PELFREY 25 2.973 2.955 0.951 0.950

> #Bahr at 21 yards - matches row #1 of data frame above

> sum(mod.fit5@beta[1] + mod.fit5@beta[2] \* 21) #Average

 kicker at 21

[1] 3.422295

> class(ranef(mod.fit5)$kicker)

[1] "data.frame"

> Bahr.kick <- row.names(ranef(mod.fit5)$kicker) ==

 "BAHR"

> sum(mod.fit5@beta[1] + mod.fit5@beta[2] \* 21) +

 ranef(mod.fit5)$kicker[Bahr.kick,]

[1] 3.320436

The REform argument corresponds to whether the random effect should be included in the calculation.

* Below is a plot of the probability of success vs. distance where a line is given for each placekicker (see program for code). The thicker red line corresponds to the “average” placekicker when bi0 = 0.



The top line corresponds to Al Del Greco who played 17 seasons in the NFL. Below is a summary of his placekicks this season:

 distance group success trials proportion

1 < 20 1 3 3 1.0000

2 >= 20 and < 30 2 33 33 1.0000

3 >= 30 and < 40 3 8 8 1.0000

4 >= 40 and < 50 4 8 9 0.8889

5 >= 50 and < 60 5 3 5 0.6000

Please note that this data set represents only about 80% of the placekicks in the NFL during the season.

The bottom line corresponds to Chip Lohmiller who played 9 seasons in the NFL with this particular season being his second to last. Below is a summary of his placekicks during this season:

 good

distance 0 1

 20 2 8

 21 0 1

 28 1 0

 29 0 1

 34 0 1

 43 1 0

 47 1 0

 52 1 0

**Sections 6.5.4 – Inference**

Wald and likelihood ratio procedures can be used for inferences. For example, the maximum likelihood estimators have large-sample normal distributions like we saw in earlier chapters. Unfortunately, both types of inference procedures can be more difficult to apply or distributional approximations less accurate than what one may expect for the sample size.

As an alternative inference approach, a parametric bootstrap approach can be performed. In Chapter 3, we implemented a parametric bootstrap approach when testing for independence in a contingency table setting. Below is the general approach that can be used for hypothesis testing when comparing a null hypothesis model to an alternative hypothesis model:

1. Calculate the test statistic of interest, say W, for the original data using the estimated model under the alternative hypothesis.
2. Generate a large number of samples (called “resamples”) using the estimated model that assumes the null hypothesis is true; these resamples should be of the same size as the original sample.
3. For each resample, estimate the model that assumes the alternative hypothesis is true and calculate the test statistic. Denote the test statistic calculated on a resample as W\*.
4. The p-value is the proportion of W\*’s which are more extreme than W. For example, if only large values of W indicate evidence against the null hypothesis, then the p-value is



where I(⋅) is an indicator function and there are B resamples.

When using the bootstrap to find a confidence interval for a parameter, the following approach can be used:

1. Generate a large number of resamples using the estimated model; these resamples should be of the same size as the original sample.
2. For each resample, estimate the model and the statistic that estimates the parameter of interest. This will lead to statistics  for b = 1, …, B.
3. These statistics provide an estimate of the probability distribution for W. Quantiles from this estimated distribution are used for a confidence interval. For example, the 95% “percentile” interval is simply the 0.025 and 0.975 quantiles from the estimated distribution. The 95% “BCa” interval is found in a similar manner, but using adjusted percentages that result in a better interval.

Example: Placekicking (PlacekickWithKickerGLMM.R, PlacekickWithNames.csv)

The anova() and Anova() functions do not work for LRTs. Below are alternative ways to test

H0: β1 = 0 (distance not in model)

Ha: β1 ≠ 0 (distance in model):

> mod.fit5.Ho <- glmer(formula = good ~ (1|kicker), nAGQ =

 5, data = kick, family = binomial(link = "logit"))

> summary(mod.fit5.Ho)

Generalized linear mixed model fit by maximum likelihood ['glmerMod']

 Family: binomial ( logit )

Formula: good ~ (1 | kicker)

 Data: kick

 AIC BIC logLik deviance

1017.4262 1027.9500 -506.7131 1013.4262

Random effects:

 Groups Name Variance Std.Dev.

 kicker (Intercept) 0 0

Number of obs: 1425, groups: kicker, 34

Fixed effects:

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

(Intercept) 2.04670 0.08323 24.59 <2e-16 \*\*\*

---

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

> #-2log(Lambda) = -2 \* (log(L\_H0) - log(L\_Ha))

> test.stat <- -2\*(logLik(mod.fit5.Ho) - logLik(mod.fit5))

> as.numeric(test.stat)

[1] 238.3341

> deviance(mod.fit5.Ho) - deviance(mod.fit5) #Equivalent

[1] 238.3341

> #p-value

> 1 - pchisq(q = as.numeric(test.stat), df = 1)

[1] 0

> #Another way

> lrt <- drop1(mod.fit5, test = "Chisq")

> lrt

Single term deletions

Model:

good ~ distance + (1 | kicker)

 Df AIC LRT Pr(Chi)

<none> 781.09

distance 1 1017.43 238.33 < 2.2e-16 \*\*\*

---

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

Testing the same hypotheses with the bootstrap:

> numb.sim <- 2000

> sim.H0 <- simulate(object = mod.fit5.Ho, nsim = numb.sim,

 seed = 8128)

> # names(sim.H0)

> head(sim.H0$sim\_1)

[1] 1 0 1 1 1 1

> head(sim.H0[[1]])

[1] 1 0 1 1 1 1

> start.time <- proc.time() #Track time to completion

> LRT.star <- numeric(length = numb.sim)

> for (i in 1:numb.sim){

 m1 <- glmer(formula = sim.H0[[i]] ~ distance +

 (1|kicker), nAGQ = 5, data = kick, family =

 binomial(link = "logit"))

 #print(i)

 LRT.star[i] <- drop1(m1, test = "Chisq")$LRT[2]

 }

> hist(x = LRT.star, xlab = "-2log(Lambda\*)", freq = FALSE)

> curve(expr = dchisq(x = x, df = 1), add = TRUE, col =

 "red")

> abline(v = test.stat, col = "blue") #Not on plot due to

 size of -2log(Lambda)



> summary(LRT.star)

 Min. 1st Qu. Median Mean 3rd Qu. Max.

 0.000001 0.118500 0.502800 1.068000 1.381000 11.370000

> mean(LRT.star >= test.stat) #p-value

[1] 0

> end.time<-proc.time()

> save.time<-end.time-start.time

> cat("\n Number of minutes running:", save.time[3]/60, "\n

 \n")

 Number of minutes running: 20.88467

With both distributional approximations, there is strong evidence that distance is important.

To compute an odds ratio for distance, the confint() function is the easiest function to use:

> exp(-10\*mod.fit5@beta[2])

[1] 3.216252

> ci.beta1.wald <- confint(object = mod.fit5, parm = 2,

 level = 0.95, method = "Wald")

> exp(ci.beta1.wald)

 2.5 % 97.5 %

distance 0.8751109 0.9046215

> rev(exp(-10\*ci.beta1.wald))

[1] 2.724777 3.796376

> ci.beta1.prof <- confint(object = mod.fit5, parm = 2,

 level = 0.95, method = "profile") # Not correct

Computing profile confidence intervals ...

> exp(ci.beta1.prof)

 2.5 % 97.5 %

(Intercept) 187.3432 722.9025

> rev(exp(-10\*ci.beta1.prof))

[1] 2.565680e-29 1.877676e-23

> ci.beta1.prof <- confint(object = mod.fit5, parm = 3,

 level = 0.95, method = "profile") # Correct

Computing profile confidence intervals ...

> rev(exp(-10\*ci.beta1.prof))

[1] 2.723266 3.839912

Comments:

1. The help file for confint.glmer() says that parm argument values correspond to an integer ordering among the β parameters in the model. Oddly, a value of 3 is needed for the correct profile LR interval to be calculated.
2. The interpretation of the odds ratio and the interval is similar to what we had in the past.
3. Due to the random effect in the model, the estimate and confidence interval are essentially for the “average” placekicker.
4. The confint() function also computes bootstrap confidence intervals, but it only appears to work when one point of quadrature is used to estimate the model. Whenever I try to use more than one point, I obtain an error message! The help file for confint.glmer() says nothing about this type of restriction. Note that this function uses the boot package to compute the intervals so one could just use the functions within the package directly to likely avoid these problems. Computational Statistics I (STAT 950) discusses how to use the boot package. I show how to compute a BCa interval without the boot package in my program. The 95% BCa interval is (2.70, 3.79).

The mcprofile profile package cannot be used to calculate intervals for parameters from a GLMM. Instead, the multcomp package can be used in a similar manner for Wald intervals. This is especially helpful for more complicated odds ratios, like those involving interactions. Note that this package was introduced in a Chapter 2 exercise.

> library(package = multcomp)

> #K <- matrix(data = c(0, -10), nrow = 1, ncol = 2) #Only

 need this one

> K <- matrix(data = c(0, -10,

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

 TRUE) #Do two rows for demonstration purposes

> K

 [,1] [,2]

[1,] 0 -10

[2,] 0 1

> linear.combo <- glht(model = mod.fit5, linfct = K)

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

 calpha = univariate\_calpha())

> ci.log.OR

 Simultaneous Confidence Intervals

Fit: glmer(formula = good ~ distance + (1 | kicker), data = kick, family = binomial(link = "logit"), nAGQ = 5)

Quantile = 1.96

95% confidence level

Linear Hypotheses:

 Estimate lwr upr

1 == 0 1.1682 1.0024 1.3340

2 == 0 -0.1168 -0.1334 -0.1002

> names(ci.log.OR)

[1] "model" "linfct" "rhs" "coef"

 "vcov" "df" "alternative" "type"

[9] "confint"

> exp(ci.log.OR$confint)

 Estimate lwr upr

1 3.2162518 2.7247767 3.7963757

2 0.8897439 0.8751109 0.9046215

attr(,"conf.level")

[1] 0.95

attr(,"calpha")

[1] 1.959964

Is a random effect really necessary? The AIC was used earlier to help make a judgment for the placekicking data set. Research in the linear mixed model setting has shown that this technique can favor the models with a random effect more than it should.

Alternatively, a formal hypothesis test or confidence interval can be constructed for the variance component. For example, a test of

H0:  = 0

Ha:  > 0

could be performed for the placekicking data example. Notice that  = 0 means there is no variability among the placekickers, so the random effect is not necessary.

The probability distribution for the estimator of a variance component tends to be highly right skewed. This causes problems with Wald-based inferences due to the large-sample normality basis for it. Also, because 0 is the lower bound for any variance, this invalidates the standard mathematical ways that the distribution for a LRT would be derived. Thus, the p-value from the χ2 distribution is incorrect. Instead, it is recommended to divide the p-value by 2.

The parametric bootstrap again provides an alternative approach. Of course, the time needed to estimate a model for each resample is a drawback. At least in a simple setting like with the placekick data, notice the null hypothesis model is just a logistic regression model without any random effect. Therefore, the timing is not a dramatic drawback for it.

With respect to confidence intervals for the variance component, similar problems occur due to  ≥ 0. Please see my book for a discussion.

Example: Placekicking (PlacekickWithKickerGLMM.R, PlacekickWithNames.csv)

LRT:

> mod.fit.wo <- glm(formula = good ~ distance, data = kick,

 family = binomial(link="logit"))

> test.stat <- -2\*(logLik(mod.fit.wo) - logLik(mod.fit5))

> test.stat

'log Lik.' 0.6530144 (df=2)

> (1 - pchisq(q = as.numeric(test.stat), df = 1))/2

[1] 0.2095185

Bootstrap test using -2log(Λ):

> numb.sim <- 2000

> sim.H0 <- simulate(object = mod.fit.wo, nsim = numb.sim,

 seed = 1722)

> LRT.star <- numeric(length = numb.sim)

> for (i in 1:numb.sim){

 mod.fit.wo <- glm(formula = sim.H0[[i]] ~ distance,

 data = kick, family = binomial(link = "logit"))

 m1 <- glmer(formula = sim.H0[[i]] ~ distance +

 (1|kicker), nAGQ = 5, data = kick, family =

 binomial(link = "logit"))

 LRT.star[i] <- -2\*(logLik(mod.fit.wo) - logLik(m1))

 }

> summary(LRT.star)

 Min. 1st Qu. Median Mean 3rd Qu. Max.

 0.0000 0.0000 0.0000 0.3788 0.2529 23.0700

> mean(LRT.star >= test.stat) #p-value

[1] 0.1595

> hist(x = LRT.star, xlab = "-2log(Lambda\*)", freq = FALSE,

 breaks = 20)

> curve(expr = dchisq(x = x, df = 1), add = TRUE, col =

 "red")

> abline(v = test.stat, col = "blue")



There is not sufficient evidence to indicate that the random effect term is needed.

Another way to perform the bootstrap test is given in the book. When implementing that approach here, the p-value is 0.1225. My program provides the code.

Final comments:

* The “choose all that apply” data setting mentioned at the beginning of these notes represents one situation where a GLMM is not appropriate. For that situation, a simple random effect term, like b0i, is used to account for the dependence among responses from a particular individual. Agresti and Liu (2001, p. 430) concisely explain that “random effects models imply independence locally, and nonnegative associations marginally” which often does not occur with this type of data.
* Dependence among responses can be handled in other way through using a generalized estimating equations (GEE) modeling approach. These models do not include random effect terms. Instead, they take into account dependence in a marginal manner. Please see Section 6.5.5 for more information.