Abraham Engle
Stat 571
January 29, 2016

1. (a) Suppose we have two parameters β_0 and β_1 and the binary outcome binary predictor model

$$\begin{aligned}&X_{ij} = j\\ &Y_{ij}|X_{ij} = x_{ij} \sim_{ind} \text{Bern}(p_{ij})\\ &p_{ij} = \text{expit}(\beta_0 + \beta_1 x_{ij})\end{aligned}$$

I fit a model using GEE in the form above, and in the the notation in the slides, we have $g = \text{expit}$, $n_i = 2$ for $i = 1, 2, \ldots, n$ and

$$X_i = \begin{pmatrix} 1 & x_{i0} \\ 1 & x_{i1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

for all $i = 1, 2, \ldots, n$. We solve the GEE

$$\sum_{i=1}^{N} \frac{\partial g(X_i)}{\partial \beta^T} V_i^{-1}(Y_i - g(X_i\beta)) = \mathbf{0}_2$$

under two working correlation matrix assumptions. First, we suppose $R_i = I_2$ for all clusters $i = 1, 2, \ldots, n$. This means

$$V_i = \phi \begin{pmatrix} S(g(\beta_0)) & 0 \\ 0 & S(g(\beta_0 + \beta_1)) \end{pmatrix}$$

since $\mu_{ij} = g(\beta_0 + \beta_1 x_{ij})$. The matrix of partial derivatives

$$D^T := \frac{\partial g(X_i)}{\partial \beta^T} = \frac{\partial}{\partial \beta^T} \begin{pmatrix} g(\beta_0) \\ g(\beta_0 + \beta_1) \end{pmatrix} = \begin{pmatrix} g'(\beta_0) & g'(\beta_0 + \beta_1) \\ 0 & g'(\beta_0 + \beta_1). \end{pmatrix}$$

Since both $\frac{\partial g(X_i)}{\partial \beta^T}$ and V_i are independent of the summation index i, we have

$$D^T V^{-1} \sum_{i=1}^{N} (Y_i - g(X_i\beta)) = \mathbf{0}_2.$$

The matrix $D^T V^{-1}$ is non-singular since $S > 0$ is a positive function of the mean and since D^T has full column rank (because $g' > 0$). Thus we reduce to solving

$$\sum_{i=1}^{N} Y_i - g(X_i\beta) = \mathbf{0}_2,$$

and since $g(X_i\beta) = \begin{pmatrix} g(\beta_0) \\ g(\beta_0 + \beta_1) \end{pmatrix}$ does not depend on i, we have the pair of equations

$$\begin{aligned}\sum_{i=1}^{N} Y_{i0} &= Ng(\beta_0)\\ \sum_{i=1}^{N} Y_{i1} &= Ng(\beta_0 + \beta_1),\end{aligned}$$

so that $\widehat{\beta}_0 = \text{logit}\left(\frac{1}{N}\sum_{i=1}^N Y_{i0}\right)$ and $\widehat{\beta}_1 = \text{logit}\left(\frac{1}{N}\sum_{i=1}^N Y_{i1}\right) - \text{logit}\left(\frac{1}{N}\sum_{i=1}^N Y_{i0}\right)$.

Suppose now that $R_i = \begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix}$, where $|\alpha| < 1$ so the inverse makes sense in the definition of the GEE. The matrix V_i is

$$V_i = \phi \begin{pmatrix} \sqrt{S(g(\beta_0))} & 0 \\ 0 & \sqrt{S(g(\beta_0 + \beta_1))} \end{pmatrix} \begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} \sqrt{S(g(\beta_0))} & 0 \\ 0 & \sqrt{S(g(\beta_0 + \beta_1))} \end{pmatrix}$$

and similarly does not depend on the cluster index $i = 1, 2, \ldots, n$ and is non-singular. Our estimating equation again looks like

$$D^T V^{-1} \sum_{i=1}^N (Y_i - g(X_i \beta)) = \mathbf{0}_2,$$

which implies

$$\sum_{i=1}^N Y_i - g(X_i \beta) = \mathbf{0}_2$$

and again $\widehat{\beta}_0 = \text{logit}\left(\frac{1}{N}\sum_{i=1}^N Y_{i0}\right)$ and $\widehat{\beta}_1 = \text{logit}\left(\frac{1}{N}\sum_{i=1}^N Y_{i1}\right) - \text{logit}\left(\frac{1}{N}\sum_{i=1}^N Y_{i0}\right)$.

(b) Now suppose we have a data generating mechanism that always gives the same number of matched pairs pairs $\{Y_{i0}, Y_{i1}\}, i = 1, 2, \ldots, 100$, with

$$\begin{aligned} X_{ij} &= j \\ Y_{ij} | X_{ij} = x_{ij} &\sim_{ind} \text{Bern}(p_{ij}) \\ p_{ij} &= \text{expit}(a_i + \beta_1 x_{ij}) \end{aligned}$$

where Y_{ij} denotes the probability of exposure for control and case ($j = 0, j = 1$ resp.) and where the intercepts $\{a_i\} = \{\Phi^{-1}((i - 0.5)/100)\}_{i=1}^{100}$ come from normal quantiles.

For my simulation, I chose β_1 between -2 and 2, incrementing by a half each time. For each β_1, I generated 10^4 independent realizations of 100 matched pairs following the distribution above and computed the estimates in part (a). Since the quantities

$$\overline{Y_{i0}}^{(s)} := \frac{1}{N}\sum_{i=1}^N Y_{i0}^{(s)}, \qquad \overline{Y_{i1}}^{(s)} := \frac{1}{N}\sum_{i=1}^N Y_{i1}^{(s)}$$

are scaled sums of i.i.d. Bernoulli random variables whose probabilities we know, by the weak law of large numbers, we know as $s \to \infty$,

$$\overline{Y_{i0}}^{(s)} \to E\left[\overline{Y_{i0}}^{(1)}\right] = \frac{1}{N}\sum_{i=1}^N P(Y_{i0} = 1) = \frac{1}{N}\sum_{i=1}^N \text{expit}(a_i) = \frac{1}{2}$$

In this simulation, we have $N = 100$ and we have $s = 1, 2, \ldots, 10^4$ replicates. Similarly, we know

$$\overline{Y_{i1}}^{(s)} \to E\left[\overline{Y_{i1}}^{(1)}\right] = \frac{1}{N}\sum_{i=1}^N P(Y_{i1} = 1) = \frac{1}{N}\sum_{i=1}^N \text{expit}(a_i + \beta_1).$$

Since β_1 does not appear in the first expression, our simulations should give $\widehat{\beta}_0 \approx \text{logit}(1/2) = 0$ for any choice of β_1 by continuity, and they do. I include the actual numbers at the end of the file, but they're all about 10^{-3}. Similarly, continuity and the weak law give

$$\widehat{\beta}_1^{(s)} = \text{logit}\left(\frac{1}{N}\sum_{i=1}^{N} Y_{i1}^{(s)}\right) - \text{logit}\left(\frac{1}{N}\sum_{i=1}^{N} Y_{i0}^{(s)}\right) \to \text{logit}\left(\frac{1}{N}\sum_{i=1}^{N} \text{expit}(a_i + \beta_1)\right) - \underbrace{\text{logit}\left(\frac{1}{N}\sum_{i=1}^{N} \text{expit}(a_i)\right)}_{=0}$$

and for my range of β_1 from -2 to 2 by half-step increments, the theoretical and experimental values are

	Experimental	Theoretical
-2	-1.6952	-1.694
-1.5	-1.260	-1.258
-1.0	-0.827	-0.832
-0.5	-0.411	-0.414
0	-0.001	0
0.5	0.416	0.414
1.0	0.831	0.832
1.5	1.255	1.258
2.0	1.694	1.694

Table 1: Experimental and Theoretical Values of $\widehat{\beta}_1$ Compared to Specified β_1 (Left Column)

(c) Our model in (a) was $E[Y_{ij}|X_{ij} = x_{ij}] = \text{expit}(\beta_0 + \beta_1 x_{ij})$, a marginal model with the same intercept across all matched pairs. However, this data generating process supposed separate intercepts for each of the i matched pairs. Similar to the example with a negative slope for hours studied marginally compared to positive slopes conditionally, we see a the bias in the estimate $\widehat{\beta}_1$ precisely because our analysis in (a) was based on a model that dealt marginally rather than conditionally. I would reply that GEE does work for matched pairs, but we need to make sure we're in agreement about the modeling assumptions before computing estimates. We came up with a different (also biased) estimate for β_1 in the case of separate intercepts based on MLE dealing with many nuisance parameters, and we also dealt with this problem last quarter using conditional MLEs.

2. My code for the Fisher Scoring algorithm is at the end of the document with the R code for the other two problems. I ran my algorithm against the geeM library in R. My stopping criterion was based on the 2-norm difference between successive iterations: $\left\|\beta^{(s+1)} - \beta^{(s)}\right\|_2 < 10^{-6}$. With this tolerance and using as a starting point the MLE outputted from GLM, my algorithm converged in two iterations for both AR-1 and exchangeable working correlation matrices. In both cases, I needed only one iteration from the initial point to reach $< 10^{-4}$ accuracy in the coordinates of $\widehat{\beta}$ outputted from the R library and one iteration to about 2×10^{-3} accuracy of $\widehat{\alpha}$ that geeM output in the AR(1) case compared to one iteration in the exchangeable case that gave me accuracy to about machine zero. My answers are

Estimate	Value
$\widehat{\beta}_0$	21.2090909
$\widehat{\beta}_1$	0.4795455
$\widehat{\beta}_2$	1.4065341
$\widehat{\beta}_3$	0.3048295
$\widehat{\alpha}$	0.6100109

Estimate	Value
$\widehat{\beta}_0$	21.1911889
$\widehat{\beta}_1$	0.4838049
$\widehat{\beta}_2$	1.5612026
$\widehat{\beta}_3$	0.2855100
$\widehat{\alpha}$	0.6193022

Table 2: Left: Exchangeable Working Correlation. Right: AR(1) Working Correlation

3. (a) In homework 2, I tried to fit the model $\boldsymbol{Y}_i = \boldsymbol{X}_i\beta$, where $\boldsymbol{X}_i$ is a 2×4 covariate matrix for the ith leprosy patient. I chose to let

$$\boldsymbol{X}_i = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1_A(i) & 1_B(i), \end{pmatrix}$$

where 1_A is the 0-1 indicator for whether patient i was given antibiotic A and where 1_B is similarly defined. This set of matrix equations models for each $i = 1, 2, \ldots, n$,

$$\begin{aligned} Y_{i1} &= \beta_0 \\ Y_{i2} &= \beta_1 + 1_A(i)\beta_2 + 1_B(i)\beta_3. \end{aligned}$$

I assume counts within the clusters (in this case, a particular patient's counts) first are independent and then unstructured. I use a Gaussian link so that $S = 1$ and R gives the following estimates:

$$\begin{aligned} \widehat{\beta}_{GLM} &= (10.733333, 12.299991, -6.999991, -6.199991)^T \\ \widehat{\beta}_{Indep} &= (10.733333, 12.299991, -6.999991, -6.199991)^T \\ \widehat{\beta}_{Unstr} &= (10.733333, 10.670200, -4.292024, -4.018575)^T \end{aligned}$$

(b) The estimates for independence are the same as GLM since we can take the estimating equation

$$\sum_{i=1}^{n} \boldsymbol{X}_i^T(\boldsymbol{Y}_i - \boldsymbol{X}_i\beta) = \mathbf{0}_4$$

and stack all the data matrices and leprosy counts contiguously and solve

$$\boldsymbol{X}_{4\times 2n}^T(\boldsymbol{Y}_{2n\times 1} - \boldsymbol{X}_{2n\times 4}\beta_{4\times 1}) = \mathbf{0}_4,$$

which is the same equation as above and is just the normal equation from OLS. When we want an unstructured working correlation, we have $V_i = \phi \begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix}$, $|\alpha| < 1$, and we have $V_i^{-1} = \phi^{-1}\begin{pmatrix} 1 & -\alpha \\ -\alpha & 1 \end{pmatrix}$, so the estimating equation is

$$\sum_{i=1}^{n} \boldsymbol{X}_i^T \phi^{-1} \begin{pmatrix} 1 & -\alpha \\ -\alpha & 1 \end{pmatrix} (\boldsymbol{Y}_i - \boldsymbol{X}_i\beta) = \mathbf{0}_4.$$

Since $\phi \neq 0$, we can multiply through and ignore it when seeking $\widehat{\beta}$. Using the formula for $\boldsymbol{X}_i$ above, our equation is (after cancelling the determinant $1 - \alpha^2$)

$$\sum_{i=1}^{n} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1_A(i) \\ 0 & 1_B(i) \end{pmatrix} \begin{pmatrix} 1 & -\alpha \\ -\alpha & 1 \end{pmatrix} (Y_i - X_i\beta) = \sum_{i=1}^{n} \begin{pmatrix} 1 & -\alpha \\ -\alpha & 1 \\ -\alpha 1_A(i) & 1_A(i) \\ -\alpha 1_B(i) & 1_B(i) \end{pmatrix} (Y_i - X_i\beta) = \mathbf{0}_4.$$

Using the definition of $\boldsymbol{X}_i$, we can look at the first two rows of the estimating equation and quickly solve for $\widehat{\beta}_0$ and verify it is the same as in the OLS case:

$$\sum_{i=1}^{n} (Y_{i1} - \beta_0) - \alpha(Y_{i2} - (\beta_1 + \beta_2 1_A + \beta_3 1_B) = 0 \tag{1}$$

$$\sum_{i=1}^{n} -\alpha(Y_{i1} - \beta_0) + (Y_{i2} - (\beta_1 + \beta_2 1_A + \beta_3 1_B)) = 0. \tag{2}$$

Adding $-(1/\alpha)$ copies of the first equation into the second, we solve

$$\left(\frac{1}{\alpha} - \alpha\right) \sum_{i=1}^{n} (Y_{i1} - \beta_0) = 0,$$

and since $|\alpha| < 1$, we get $\beta_0 = \overline{Y_{i1}}$ as before. What's left is three linear equations in three unknowns depending on the fourth parameter α.

R Code

```
setwd("~/Dropbox/UW2015-2016/Win2016/571/hw3")
library(ggplot2)
library(geeM)
library(reshape2)
library(boot)
##prob1
a = qnorm(ppoints(100))
g <- function(x)
{
  return(
    exp(x)/(1+exp(x))
  )
}
n = 10000
X = cbind(rep(1,2),c(0,1))
beta1 = seq(-2,2,0.5)
b1hat = rep(0,length(beta1))
b0hat = b1hat
trueb1hat = b1hat
trueb0hat = b1hat
medianDisc = b1hat
set.seed(1)
for(k in 1:length(beta1))
{
  simulation = matrix(0,nrow=n,ncol=2)
  discpairs = rep(0,n)
  for(i in 1:n){
    N = 100 #total number of matched pairs
    Y = matrix(0,nrow=100,ncol=2)
    m01 = 0 #counts for discordant pairs
    m10 = 0
    for(j in 1:N)
    {
      beta = c(a[j],beta1[k])
      Y[j,1] = rbinom(1,1,g(X%*%beta)[1])
      Y[j,2] = rbinom(1,1,g(X%*%beta)[2])
      if(Y[j,1] == 0 & Y[j,2] == 1)
      {
        m01 = m01 + 1
      }
      if(Y[j,1] == 1 & Y[j,2] == 0)
      {
        m10 = m10 + 1
      }
    }
  discpairs[i] <- m01/m10
  simulation[i,] <- colMeans(Y)
  }

  trueb0hat[k] = logit(mean(g(a)))
  trueb1hat[k] = logit(mean(g(a+beta1[k]))) - logit(mean(g(a)))

  cat("Beta1 is ", beta1[k], "\n")
  cat("Median of Discordant Pairs is ", median(log(discpairs)), "\n")
  medianDisc[k] = median(log(discpairs))
  cat("First Estimate is ", logit(mean(simulation[,1])), "\n")
  b0hat[k] = logit(mean(simulation[,1]))
  cat("Second Estimate is ", logit(mean(simulation[,2]))-logit(mean(simulation[,1])), "\n")
  b1hat[k] = logit(mean(simulation[,2]))-logit(mean(simulation[,1]))
}

plot(beta1,trueb0hat,ylim=c(-0.01,0.01),col="blue")
points(beta1,b0hat,col="red")

plot(beta1,abs(trueb1hat-b1hat),col="blue")
points(beta1,b1hat,col="red")


hist(simulation[,1],main=expression(paste("Histogram of Estimates ", hat(beta)[0])))
abline(v=mean(g(a)),col="red", lty=4, lwd=3)

hist(simulation[,2], main=expression(paste("Histogram of Case Exposure Proportions")))
abline(v=mean(g(a+beta1)),col="red", lty=4, lwd=3)

hist(logit((simulation[,2]))-logit((simulation[,1])),main=expression(paste("Histogram of Estimate
abline(v=logit(mean(g(a+beta1)))-logit(mean(g(a))),col="red", lty=4, lwd=3)

##########################
###estimates of b0hat#####
#> b0hat
#[1] -0.001900001  0.000344000 -0.002124001  0.001536000  0.001520000  0.000480000
#[7]  0.000428000 -0.001200000 -0.000772000
##########################
##########################




(1-g(a))*(g(a+beta1[1]))
#all probabilities are <0.07
#beta1 = -2
#this is the probability of (0,1) goes into m01 count
```

```
(g(a))*(1-g(a+beta1[length(beta1)]))
#all probabilities are <0.07
#beta1 = 2
#this is the probability of (1,0), goes into m10 count


#########
##prob2##
#########
library(nlme)

data(Orthodont, package="nlme")

d4 <- Orthodont # using shorter names
d4$id <- d4$Subject
d4$male <- d4$Sex=="Male"

m1 <- glm(distance~I(age-8)*male,data=d4)
beta = m1$coefficients
betaold = beta
X = model.matrix(m1)

workcor = "AR1"
count = 0
repeat{
  count = count + 1
  phi = (1/(4*27-4))*sum((d4$distance - X%*%beta)^2)
  pearsonResid = phi^(-1/2)*(d4$distance - X%*%beta)

  pearson_resid = matrix(pearsonResid,nrow=27,ncol=4,byrow=TRUE)

  start = 4*0:26+1
  n = 27
  ni = 4
  p = 4

  if(workcor == "AR1"){
  alphaAR1 = 0
    for(i in 1:n){
      for(j in 1:(ni-1)){
        alphaAR1 <- alphaAR1 + pearson_resid[i, j] * pearson_resid[i, j+1]
      }
    }
    alphaAR1 <- 1 / (n*ni - n - p) * alphaAR1
```

```
}


if(workcor == "Exchangeable")
{
alphaExch = 0
for(i in 1:n)
{
  for(j in 1:(ni-1))
  {
    for(k in (j+1):ni)
    {
      alphaExch <- alphaExch + pearson_resid[i, j] * pearson_resid[i, k]
    }
  }
}
alphaExch <- 1 / (n / 2 * ni * (ni - 1) - p) * alphaExch
}
##################################
##matrices Ri do not vary over i##
##################################
RiExch = matrix(alphaExch,nrow=4,ncol=4)
diag(RiExch) <- rep(1,4)

RiAR1Inv = matrix(0,nrow=4,ncol=4)
diag(RiAR1Inv) = 1 + alphaAR1^2
RiAR1Inv[1,1] = 1
RiAR1Inv[4,4] = 1
RiAR1Inv[row(RiAR1Inv)==(col(RiAR1Inv)+1)] = -alphaAR1
RiAR1Inv[(row(RiAR1Inv)+1)==col(RiAR1Inv)] = -alphaAR1
RiAR1Inv = RiAR1Inv/(1-alphaAR1^2)

ViInvAR1 = (1/phi)*RiAR1Inv
ViInvExch = (1/phi)*solve(RiExch)

HessAR1 = matrix(0,nrow=4,ncol=4)
gradAR1 = matrix(0,nrow=4,ncol=1)
HessExch = matrix(0,nrow=4,ncol=4)
gradExch = matrix(0,nrow=4,ncol=1)
for(i in 1:27)
{
  Di = X[start[i]:(start[i]+3),]

  HessAR1 = HessAR1 + t(Di)%*%ViInvAR1%*%Di
  gradAR1 = gradAR1 + t(Di)%*%ViInvAR1%*%(d4$distance[start[i]:(start[i]+3)] - Di%*%beta)
```

```
    HessExch = HessExch + t(Di)%*%ViInvExch%*%Di
    gradExch = gradExch + t(Di)%*%ViInvExch%*%(d4$distance[start[i]:(start[i]+3)] - Di%*%beta)
  }
  if(workcor == "Exchangeable"){
  beta = beta + solve(HessExch)%*%gradExch}
  else{
  beta = beta + solve(HessAR1)%*%gradAR1
  }
  if(sum((beta-betaold)^2) < 10^(-6))
  {
    break
  }
  betaold = beta
}

alphaExch - gee2$alpha

alphaAR1 - gee1$alpha

gee1 <- geem(distance~I(age-8)*male, id=id, data=d4, corstr="ar1")
gee2 <- geem(distance~I(age-8)*male, id=id, data=d4, corstr="exchangeable")




#########
##prob3##
#########
lep <- read.table("leprosy.txt", header=TRUE, sep=" ")
attach(lep)
n = dim(lep)[1]
ni = 2 #same for all i
p = 4 #four parameters

lep['severe']<-NULL
lep['subject'] = 1:30
lep_Long = melt(lep, id.vars = c("subject", "trt"))
lep_Long = lep_Long[order(lep_Long$subject),]
lep_Long['pre'] <- rep(c(1,0),30)
lep_Long['post'] <- rep(c(0,1),30)
lep_Long['A'] = as.numeric(lep_Long$trt==1 & lep_Long$post==1)
lep_Long['B'] = as.numeric(lep_Long$trt==2 & lep_Long$post==1)
gee1 <- gee(value~-1+pre+post+A+B, id=subject, data=lep_Long, corstr="independence")
gee2 <- gee(value~-1+pre+post+A+B, id=subject, data=lep_Long, corstr="unstructured")
summary(gee1)
```