Matthew Draper | POLI 271 | Homework #3 | April 23rd 2019

1a. $X \sim f_b(x; n, p)$; $\theta = (p)$ (n is known)

X is binomially distributed i.i.d. with mean n and variance p. Since the binomial distribution sums repeated Bernoulli trials, we can omit product notation in our likelihood expression:

$\mathcal{L}(p) = (^{n}_{k})\, p^k\, (1 - p)^{n-k}$

$\mathcal{L}(p) = (n! / k!(n - k)!)\, p^k\, (1 - p)^{n-k}$

$\log \mathcal{L}(p) = \log(n! / k!(n - k)!) + \log(p^k) + \log((1 - p)^{n-k})$

$\log \mathcal{L}(p) = \log(n! / k!(n - k)!) + k*\log(p) + (n-k)*\log(1 - p)$

1b. $S(p) = \nabla \log \mathcal{L}(p) = \partial/\partial p \log \mathcal{L}(p)$

$\partial/\partial p \log \mathcal{L}(p) = \partial/\partial p \log(n! / k!(n - k)!) + k*\log(p) + (n-k)*\log(1 - p) =$

$= 0 + k/p + (n - k)/(1 - p)$

1c. $\hat{p}$: $0 = k/p + (n - k)/(1 - p)$

$k(1 - p) = p(n - k)$

$k - kp = pn - pk$

$k = pn$

$\hat{p} = k/n$

1d. $I(\hat{p}) = -\partial/\partial p\, S(p)\,|_{\hat{p}}$

$S(p) = k/p + (n - k)/(1 - p)$

$-\partial/\partial p\, k/p + (n - k)/(1 - p)\,|_{\hat{p}} = k / (k/n)^2 + (n - k)/(1-k/n)^2$

$= (k/(k^2/n^2)) + (n - k)/((n/n) - (kn/n))^2$

$= ((n - k)n^2 + kn^2)/ k(n - k)$

$= n^3 / k(n - k)$

1e. $I(\hat{p}) = n^3 / k(n - k)$; $k = \Sigma x_i$; $E(k) = np$

$E(I(\hat{p})) = n^3 / np(n - np) = n^2 / p(n - np) = n / p(1-p)$

Draper HW 3

Matthew Draper

April 22, 2019

I did my best on this problem set, but I had to cap my work after about 20 hours so I wasn't able to completely finish #3. I'm aware this is taking me longer than it would take a typical student, and I'm trying to improve my speed. I've read chapters 1-4 three times so far, and the material is starting to stick, but I have a lot of questions.

Most importantly, it seems that the diagnostic tools rely on manipulations of the log likelihood, and I would like to review how to use the outputs of GLM as inputs in the diagnostics. We keep emphasizing that MLE evaluated at a point is a single number, but the output of GLM looks a lot more like the regression output from OLS, and I'm not sure what "the MLE" is when I'm looking at a GLM BUTON. I learn best from worked examples, and I've been reviewing the worked examples in the text, but it would be helpful to do some actual problems in class.

Problem #2:

```
library(foreign);  library(car); library(arm)
```

```
## Warning: package 'car' was built under R version 3.5.2
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 3.5.2
```

```
## Warning: package 'arm' was built under R version 3.5.2
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.5.3
```

```
## Loading required package: Matrix
```

```
## Loading required package: lme4
```

```
## Warning: package 'lme4' was built under R version 3.5.2
```

```
## 
## arm (Version 1.10-1, built: 2018-4-12)
```

```
## Working directory is C:/Users/mwdra/Google Drive/UCSD/Courses/271 Advanced Statistical Applications/Problem
Set 3
```

```
## 
## Attaching package: 'arm'
```

```
## The following object is masked from 'package:car':
## 
##     logit
```

```
mroz <- read.dta("binlfp2.dta")
lfpbin1 <- as.numeric(mroz$lfp == "inLF")
```

```
lfp <- mroz
colnames(lfp)<-c("LFP","young.kids","school.kids","age","college.woman", "college.man", "wage", "income")
lfp$lfpbin <-rep(0,length(lfp$LFP) )
lfp$lfpbin[lfp$LFP=="inLF"]<-1
attach(lfp)
x<-cbind(young.kids,school.kids,age,as.numeric(college.woman)-1,wage)
y<-lfpbin
```

```
m1<-lm(lfpbin~age+college.man+income,data=lfp)
mod.mat<-m1$model
mod.mat$college.man<-recode(mod.mat$college.man,"'College'=1; 'NoCol'=0 ",as.factor=F)
use.vars<-names(m1$model)
y<-as.vector(mod.mat$lfpbin)
x<-as.matrix(mod.mat[,2:ncol(mod.mat)])
binreg<- function(X,y,method="BFGS"){
  X<- cbind(1,X)
  negLL<- function(b,X,y){
    p<-as.vector(1/(1+exp(-X %*% b)))
    - sum(y*log(p) + (1-y)*log(1-p))
  }
  gradient<- function(b,X,y){
    p <- as.vector(1/(1+exp(-X %*% b)))
    - apply(((y - p)*X),2,sum)
  }
  results<- optim (rep(0,ncol(X)),negLL,gr=gradient,
                   hessian=T,method=method,X=X,y=y)
  list(coefficients=results$par,var=solve(results$hessian),
       deviance=2*results$value,
       converged=results$convergence==0)
}
## Coefficients
mlebin.fit<-binreg(x,y)
round(mlebin.fit$coefficients,5)
```

```
## [1]  1.28132 -0.01464  0.41489 -0.02690
```

```
## Standard Errors
stderror<-rbind(sqrt(mlebin.fit$var[1,1]),sqrt(mlebin.fit$var[2,2]),sqrt(mlebin.fit$var[3,3]),sqrt(mlebin.fit$v
ar[4,4]))
round(stderror,5)
```

```
##         [,1]
## [1,] 0.42599
## [2,] 0.00943
## [3,] 0.16900
## [4,] 0.00731
```

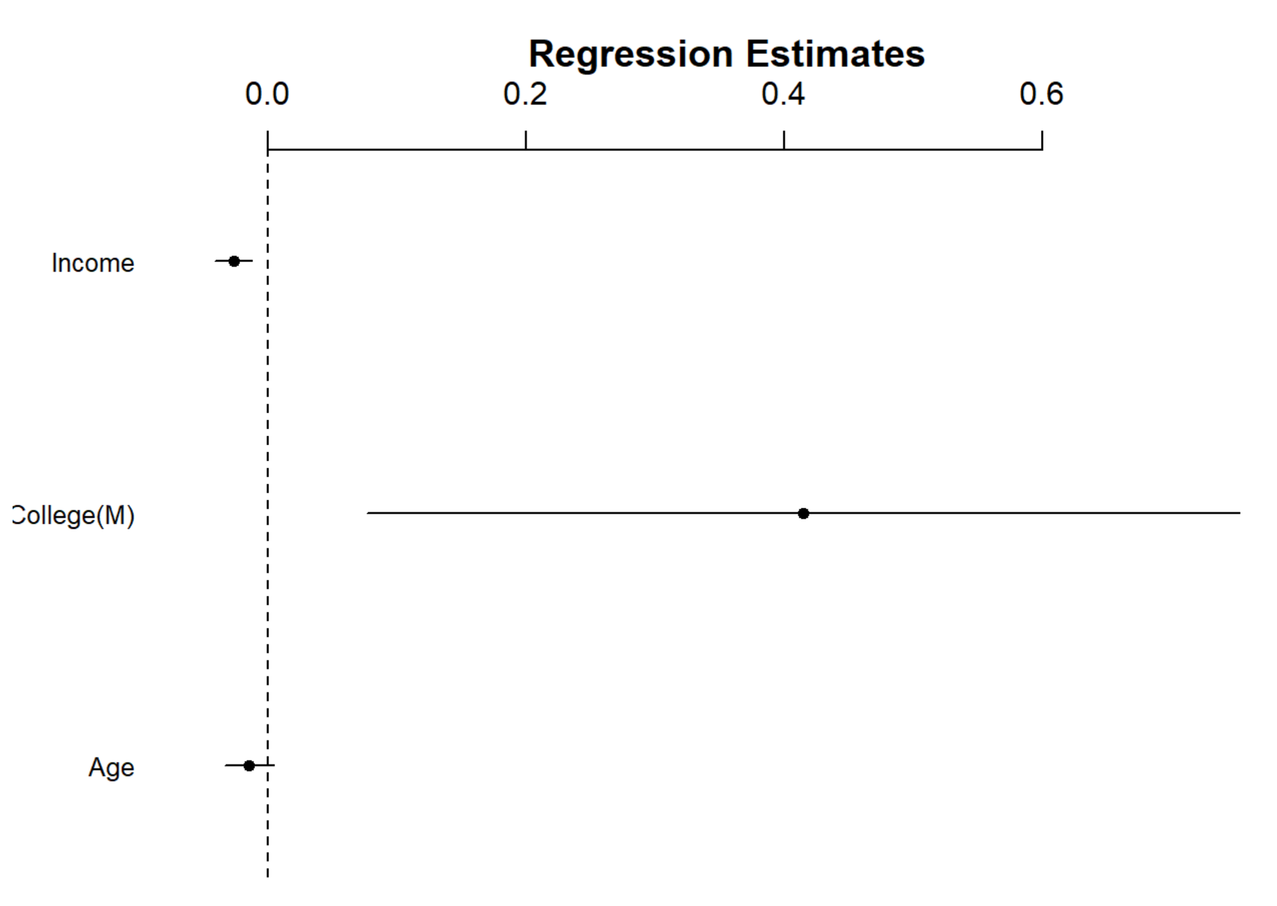
```
## AIC = -2logL + 2k; k = 3
#-2*mlebin.fit$coefficients+2*3
#extractAIC(mlebin.fit$coefficients,k = 3)

## BIC = -2logL + k log(n); k = 3
#extractBIC(mlebin.fit$coefficients,k = 3)

## AIC and BIC should both be simple calculations, but I don't understand how to get the log likelihood out of
 the GLM output so I can't get the inputs for the calculations here.

## Combined Table
cbind(mlebin.fit$coefficients,stderror)
```

```
##             [,1]        [,2]
## [1,]  1.28131714 0.425993613
## [2,] -0.01463531 0.009425953
## [3,]  0.41488547 0.169002357
## [4,] -0.02690202 0.007313083
```

```
fit.glm<-glm(lfpbin~age+college.man+income,
family = binomial (link = "logit"), data = lfp)
summary(fit.glm)
```

```
## 
## Call:
## glm(formula = lfpbin ~ age + college.man + income, family = binomial(link = "logit"), 
##     data = lfp)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.6814  -1.2464   0.9056   1.0509   1.7592  
## 
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)    
## (Intercept)         1.281317   0.425935   3.008 0.002628 ** 
## age                -0.014635   0.009424  -1.553 0.120417    
## college.manCollege  0.414885   0.169003   2.455 0.014092 *  
## income             -0.026902   0.007313  -3.679 0.000234 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance: 1009.1  on 749  degrees of freedom
## AIC: 1017.1
## 
## Number of Fisher Scoring iterations: 4
```

```
lcb<-coef(fit.glm)-2*sqrt(diag(vcov(fit.glm)))
ucb<-coef(fit.glm)+2*sqrt(diag(vcov(fit.glm)))
lcb<-lcb[-1]
ucb<-ucb[-1]
coefplot(fit.glm, varnames=c("Intercept","Age","College(M)","Income"),
        cex.pts=1.1, lwd=2, lower.conf.bounds=lcb, offset=0.4,
        upper.conf.bounds=ucb, add=F)
```



We observe that income is highly significant, and that college (man) is moderately significant. The significance of the intercept terms imply that our new model is picking up on an underlying data-generating process. However, I don't see any reason to prefer this model to the model given by table 3.5 in the text. Our new model has the virtue of using fewer predictor variables, but we still end up with a higher AIC than the model in table 3.5 (1017.1>937). Since lower AIC scores are better, we prefer the initial model. In addition, the p-values (and z-scores) are lower in Table 3.5.

Problem #3:

Congressional articles of impeachment are separate resolutions, each triggering a trial in the Senate if passed, so I will consider a "vote for the impeachment of President Clinton in 1999" as a vote for one or more articles of impeachment. Since articles of impeachment require majorities to pass, abstention is functionally identical to a no vote.

```
imp<-read.csv("impeach.csv")
imp$votesum[is.na(imp$votesum)] <- 0
imp$yea<-as.numeric(imp$votesum>0)
```

```
imp.glm<-glm(yea ~ clint96 + partyid,
family = binomial (link = "logit"), data = imp)
summary(imp.glm)
```

```
## 
## Call:
## glm(formula = yea ~ clint96 + partyid, family = binomial(link = "logit"), 
##     data = imp)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -3.2682  -0.0411   0.0126   0.0909   2.5772  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) 11.26044    3.25982   3.454 0.000552 ***
## clint96     -0.31619    0.07558  -4.184 2.87e-05 ***
## partyid      7.98766    1.03234   7.737 1.01e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 601.821  on 434  degrees of freedom
## Residual deviance:  57.887  on 432  degrees of freedom
## AIC: 63.887
## 
## Number of Fisher Scoring iterations: 9
```

```
imp.dem<-subset(imp, imp$partyid==0)
imp.rep<-subset(imp, imp$partyid==1)

imp.glm.dem<-glm(yea ~ clint96 + partyid,
family = binomial (link = "logit"), data = imp.dem)
summary(imp.glm.dem)
```

```
## 
## Call:
## glm(formula = yea ~ clint96 + partyid, family = binomial(link = "logit"), 
##     data = imp.dem)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -1.81000  -0.13707  -0.03001  -0.00599   2.66502  
## 
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)   
## (Intercept)  13.9721     5.0115   2.788  0.00530 **
## clint96      -0.3803     0.1184  -3.212  0.00132 **
## partyid           NA         NA      NA       NA   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 47.111  on 206  degrees of freedom
## Residual deviance: 24.571  on 205  degrees of freedom
## AIC: 28.571
## 
## Number of Fisher Scoring iterations: 9
```

```
imp.glm.rep<-glm(yea ~ clint96 + partyid,
family = binomial (link = "logit"), data = imp.rep)
summary(imp.glm.rep)
```

```
## 
## Call:
## glm(formula = yea ~ clint96 + partyid, family = binomial(link = "logit"), 
##     data = imp.rep)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -3.12084   0.05831   0.10946   0.18086   0.85279  
## 
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept)  15.9660     5.2593   3.036   0.0024 **
## clint96      -0.2524     0.1029  -2.451   0.0142 * 
## partyid           NA         NA      NA       NA  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 40.274  on 227  degrees of freedom
## Residual deviance: 32.628  on 226  degrees of freedom
## AIC: 36.628
## 
## Number of Fisher Scoring iterations: 8
```

```
imp2.glm<-glm(yea ~ clint96 + partyid + ccoal98,
family = binomial (link = "logit"), data = imp)
summary(imp2.glm)
```

```
## 
## Call:
## glm(formula = yea ~ clint96 + partyid + ccoal98, family = binomial(link = "logit"), 
##     data = imp)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -3.02500  -0.03478   0.01244   0.05919   2.93420  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  5.00826    4.15596   1.205  0.22817    
## clint96     -0.22280    0.08962  -2.486  0.01292 *  
## partyid      5.78123    1.12023   5.161 2.46e-07 ***
## ccoal98      0.05584    0.01774   3.147  0.00165 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 599.247  on 432  degrees of freedom
## Residual deviance:  45.233  on 429  degrees of freedom
##   (2 observations deleted due to missingness)
## AIC: 53.233
## 
## Number of Fisher Scoring iterations: 9
```

```
## Likelihood Ratio: LR(thetaR,thetaG|x) = -2log(L(thetaR)|x)/(L(thetaG)|x))
```

I have no idea how to do this. The sticking point is extracting "the maximum likelihood" from the output of GLM. Performing the ratio operation seems fairly straightforward, because our null hypothesis (restricted model = correct) predics a chi-squared distribution of the likelihood ratio.

```
predicted<-fitted(imp.glm)
actual<-as.vector(imp2.glm)
pred<-prediction(predicted,actual)
```

```
## Error in prediction(predicted, actual): could not find function "prediction"
```

```
perf<-performance(pred,"tpr","fpr")
```

```
## Error in performance(pred, "tpr", "fpr"): could not find function "performance"
```

```
predsimp<-prediction(pred.simp,actual)
```

```
## Error in prediction(pred.simp, actual): could not find function "prediction"
```

```
perf.simp<-performance(predsimp,"tpr","fpr")
```

```
## Error in performance(predsimp, "tpr", "fpr"): could not find function "performance"
```

```
par(las=1, bty="n")
plot(perf,main="ROC plots for competing models", bty="n",lwd=2)
```

```
## Error in plot(perf, main = "ROC plots for competing models", bty = "n", : object 'perf' not found
```

```
plot(perf.simp, lwd=2, add=T)
```

```
## Error in plot(perf.simp, lwd = 2, add = T): object 'perf.simp' not found
```

```
lines(actual,actual, lty="dashed")
```

```
## Error in xy.coords(x, y): (list) object cannot be coerced to type 'double'
```

I'm really trying, but apparently package ROCR isn't compatible with my version of R (which also appears to be the latest version), so I can't get any functionality out of the "performance" and "prediction" functions. I spent a couple of hours on Stack Exchange trying to find a workaround, but eventually I just had to call time and stop. I'll come to office hours this week and we can discuss what I'm doing wrong.