Generalized Linear Models

Scott Creel

Wednesday, September 10, 2014

This exercise extends the prior material on using the lm() function to fit an OLS regression and test hypotheses about effects on a parameter. You can extend your use of regression models by considering **Generalized Linear Models** or **GLMs**. The **glm()** function accomplishes the same basic tasks as lm(), but it is more flexible.

The **lm()** function assumes that the residuals are normally distributed and there is a one-to-one or identity 'link' between Y and X. **glm()** allows for other distributions and links. Three of the most important are:

family = gaussian(link = "identity") - Same as OLS regression. Appropriate for normally distributed dependent variables

family = binomial(link = "logit") - Appropriate for dependent variables that are binomially distributed such as survival (lived vs died) or occupancy (present vs absent)

family = poisson(link = "log") - Appropriate for dependent variables that are counts (integers only) such as group size

Here is a brief example to compare glm() and lm(), showing why we need generalized linear models when the dependent variable does not meet the assumptions of OLS: Let's test whether home range quality affects survival

- survive is a binomial variable
- survive = 1 means the individual lived
- survive = 0 means the individual died
- home.range.quality is the ranked order of home ranges in terms of resources available to the individual

```
survive <- c(0,0,0,0,1,0,1,1,1,1,1)
home.range.quality <- seq(0,1,0.1)
```

Fit an **inappropriate** OLS linear model using lm()

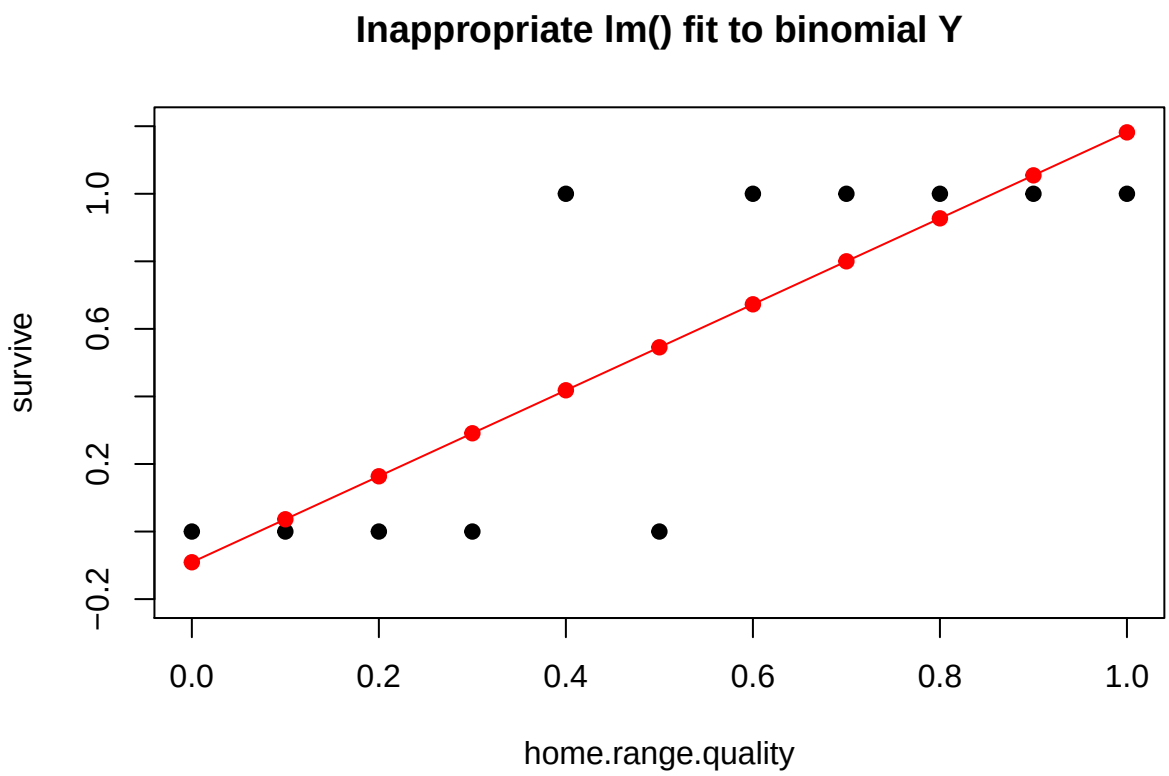
```
mod4b <- lm(survive ~ home.range.quality)
summary(mod4b)
```

```
##
## Call:
## lm(formula = survive ~ home.range.quality)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5455 -0.1727 -0.0364  0.1455  0.5818
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.0909     0.1828   -0.50   0.6309
## home.range.quality   1.2727     0.3090    4.12   0.0026 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.324 on 9 degrees of freedom
## Multiple R-squared:  0.653,  Adjusted R-squared:  0.615
## F-statistic:   17 on 1 and 9 DF,  p-value: 0.0026
```

Incorrectly using lm() to fit an OLS regression model a binomial dependent variable, the summary makes the results look pretty good: the effect of home range quality on survival is large, with a very small P-Value (0.0026) and a coefficient of determination of 61.5%. Looks like a strong effect, but when you plot the fitted model and the data, important problems are revealed.

```
plot(home.range.quality, survive, pch = 19, ylim = c(-0.2, 1.2),
     main = 'Inappropriate lm() fit to binomial Y')
points(home.range.quality,fitted(mod4b),col=2, pch = 19)
lines(home.range.quality,fitted(mod4b),col=2)
```



There are several obvious problems with the lm().

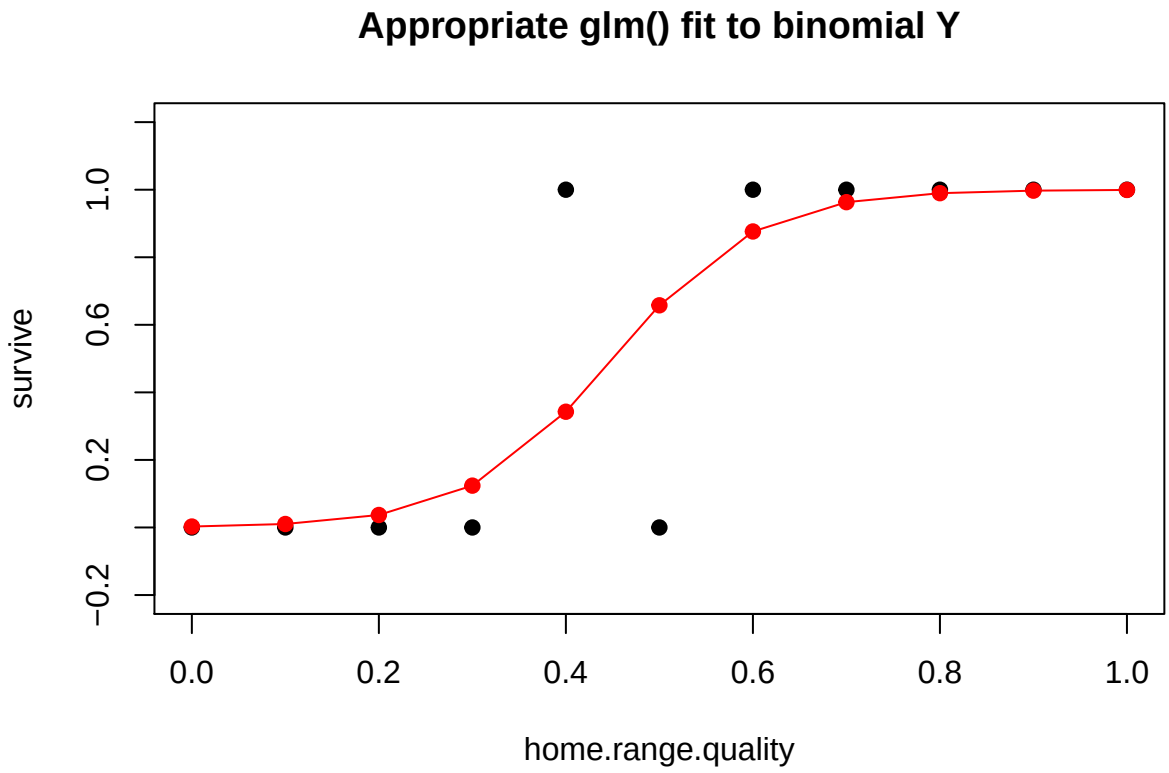
1. The model doesn't fit very well for points in the middle of the x-axis.
2. For some home ranges, the model estimates survival rates less than zero and greater than one, which are impossible. This is the bigger problem – we can't use a model that predicts survival rates greater than 100% or a chance of negative survival!

in reality, survival is binomially distributed (each individual either lives, 1, or dies, 0), and 'bounded' in the interval [0,1], meaning that survival can never be less than zero or greater than one. Fitting a more-appropriate

generalized linear model with glm() that correctly specifies that survival is a binomial variable helps with both of these problems.

```
mod4a <- glm(survive ~ home.range.quality, family = binomial)
```

```
plot(home.range.quality, survive, pch = 19, ylim = c(-0.2, 1.2),
     main  = 'Appropriate glm() fit to binomial Y')
#note we're using the glm results now, mod4a
points(home.range.quality,fitted(mod4a),col=2, pch = 19)
lines(home.range.quality,fitted(mod4a),col=2)
```



Now we see a regression model has been fit that is bounded to remain between 0 and 1, just like the data. **The model must match the structure of the data to provide valid inferences.**

The model that is fit when you run

```
glm(y~x, family = binomial)
```

assumes a default 'link function' of logit, so it's equivalent to

```
glm(y~x, family = binomial(link = 'logit'))
```

But what does this mean? Logit is just statisticians' shorthand for 'log-odds-ratio'. If S is the proportion of individuals that survive, then logit(s) is

$$logit(s) = ln\left(\frac{s}{1-s}\right)$$

Within parenthesis, the fraction is the **odds-ratio** or in this case, the **odds of survival**. Suppose that 80% of individuals survive, so s = 0.8. $\frac{s}{(1-s)}$ is 0.8/0.2, indicating an indivual is 4 times more likely to live than to die. The logit is just the log of the odds ratio. The regression model being fit by glm() is

$$logit(s) = ln\left(\frac{s}{1-s}\right) = \beta_0 + \beta_1 X$$

The **log-odds-of-survival** is a linear function of home range quality in this model. The survival rate itself is **not** a straight-line function of hime range quality, because of the link function.

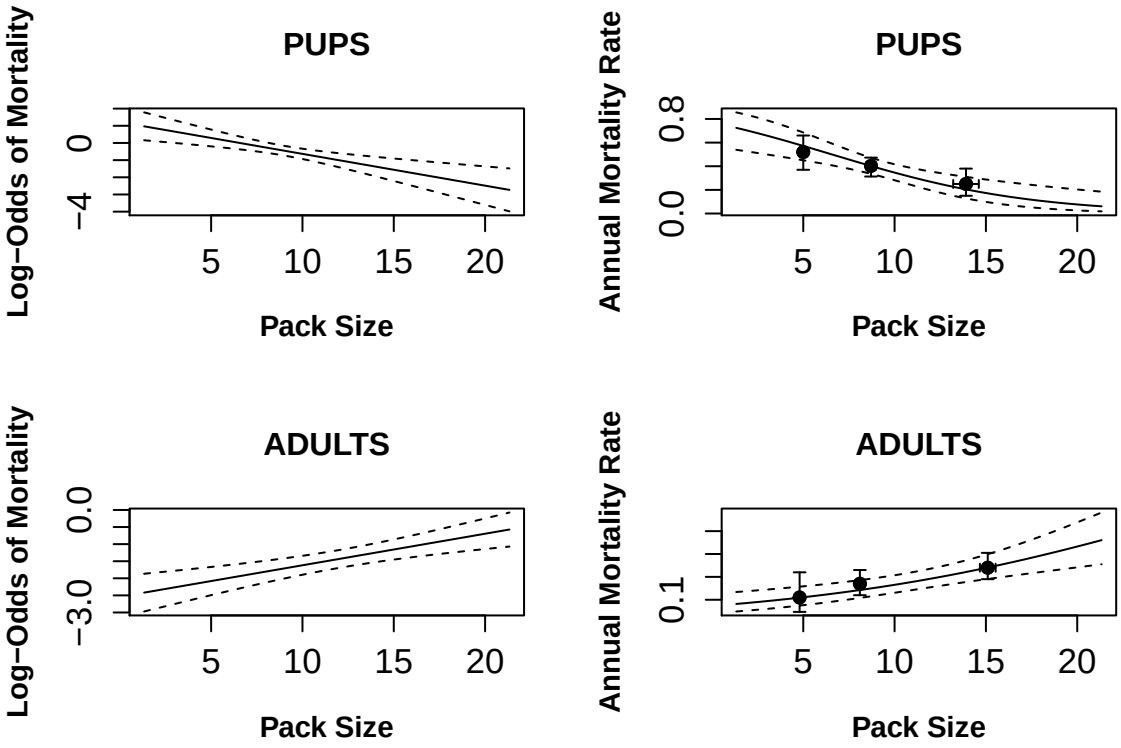
Looking at a summary of the glm:

```
summary(mod4a)
```

```
##
## Call:
## glm(formula = survive ~ home.range.quality, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4644  -0.2092   0.0391   0.2090   1.4635
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -5.87       3.94   -1.49     0.14
## home.range.quality    13.05       8.36    1.56     0.12
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15.1582  on 10  degrees of freedom
## Residual deviance:  5.0196  on  9  degrees of freedom
## AIC: 9.02
##
## Number of Fisher Scoring iterations: 7
```

You must keep in mind that the regression coefficient in the summary of this glm() is on the logit scale. With a little reflection this shoulf be clear: when you look at the regression output, the effect of home range quality on survival is 13.05 +/- a standard error of 8.36. But when you look at the plotted results, the relationship is not a straight line, as a single, constant slope would imply. The coefficient means that **log-odds of survival changes by 13.05 for every unit of change in home range quality**. You have to **back-transform** from the logit-scale if you want to see the results on the original scale.

In practice this is a bit complex, but the plot below shows the original GLM on the left and the back-transformed model on the right, for a real example.



This should give you a working understanding of the reasons that we use generalized linear models and how to use the output of the glm() function.

I'll append the code for the last example below, but I *do not expect you to work through it for this class.* It requires the package **plotrix**.

```
# Plotting  Mortality Rates from Logistic Regressions


#  partial derivatives of log odds
#  with respect to b0 & b1 respectively
#  from logistic regression equations

library(plotrix)

d0 <- D(expression(b0+b1*packsize),"b0")
d1 <- D(expression(b0+b1*packsize),"b1")

# note that as written, this is for a regression
# coded Die=1, Live=0

#enter logistic regression coefficients for pups
b0 <- 1.2152
b1 <- -0.1851
```

```
#enter variance-covariance matrix from logistic regression for pups
VarCov <- matrix(0,nrow=2,ncol=2)
VarCov[1,] <- c(0.22832, -0.02298)
VarCov[2,] <- c(-0.02298, 0.002546)

CIlnOdds <- matrix(0,nrow=61,ncol=4)
CIS <- matrix(0,nrow=61,ncol=4)

for(i in 1:61) {
packsize <- 1 + i/3

der <- matrix(0,nrow=2,ncol=1)
der[1,1] <- eval(d0)
der[2,1] <- eval(d1)

# to get the Variance of the log-odds for a given set of covariate
# values, multiply a row vector of derivatives
# by the Var-Covar matrix by a column vector of derivatives

VarLnOdds <- t(der)%*%VarCov%*%(der)
SELnOdds <- sqrt(VarLnOdds)
LnOdds <- b0+b1*packsize
LCIlnOdds <- LnOdds - 1.96*SELnOdds
UCIlnOdds <- LnOdds + 1.96*SELnOdds
CIlnOdds[i,1] <- LCIlnOdds
CIlnOdds[i,2] <- LnOdds
CIlnOdds[i,3] <- UCIlnOdds
CIlnOdds[i,4] <- packsize
S <- exp(LnOdds)/(1+exp(LnOdds))
LCIS <- exp(LCIlnOdds)/(1+exp(LCIlnOdds))
UCIS <- exp(UCIlnOdds)/(1+exp(UCIlnOdds))
CIS[i,1] <- LCIS
CIS[i,2] <- S
CIS[i,3] <- UCIS
CIS[i,4] <- packsize
}

# Plot the log-odds for mortality
par(mfrow=c(2, 2),oma=c(0, 1, 2, 1))
x <- CIlnOdds[,4]
y <- CIlnOdds[,1:3]
matplot(x,y,type="l",lty=2:1, col=1,
ps = 14,
xlab = "Pack Size",
ylab = "Log-Odds of Mortality", cex.lab = 1.1, cex.axis = 1.3, font.lab = 2,
main = "PUPS")

# Plot the estimated mortality rate
x <- CIS[,4]
y <- CIS[,1:3]
matplot(x,y,type="l", lty=2:1, col=1,
xlab = "Pack Size",
ylab = "Annual Mortality Rate", cex.lab = 1.1, cex.axis =1.3, font.lab = 2,
```

```
main = "PUPS")

#add means for S, M, L packs with error bars in x and y

packsize <- c(5.0, 8.71, 13.91)
mortmean <- c(0.520, 0.403 ,0.250)
mortupper95 <- c(0.14, 0.07, 0.13)
mortlower95 <- c(0.15, 0.09, 0.10)

packsizeSD <- c(0.0 ,1.51 ,2.86)
n<- c(27,129,64)
packsize95 <- (1.96*(packsizeSD/sqrt(n)))

attach(plotCI)

plotCI(packsize, mortmean, err="y", uiw = mortupper95,
liw = mortlower95, pch = 19, add=TRUE)

plotCI(packsize, mortmean, err="x", uiw = packsize95,
liw = packsize95, pch = 19, add=TRUE)


########################################
# Now repeat the whole exercise for Adults

#  partial derivatives of log odds
#  with respect to b0 & b1 respectively

d0 <- D(expression(b0+b1*packsize),"b0")
d1 <- D(expression(b0+b1*packsize),"b1")


#enter logistic regression coefficients for pups
b0 <- -2.54226
b1 <- 0.09232

#enter variance-covariance matrix from logistic regression for pups
VarCov <- matrix(0,nrow=2,ncol=2)
VarCov[1,] <- c(0.095732, -0.006488)
VarCov[2,] <- c(-0.006488, 0.000540)

CIlnOdds <- matrix(0,nrow=61,ncol=4)
CIS <- matrix(0,nrow=61,ncol=4)

for(i in 1:61) {
packsize <- 1 + i/3

der <- matrix(0,nrow=2,ncol=1)
der[1,1] <- eval(d0)
der[2,1] <- eval(d1)

# to get the Variance of the log-odds for a given set of covariate
# values, multiply a row vector of derivatives
```

```
# by the Var-Covar matrix by a column vector of derivatives

VarLnOdds <- t(der)%*%VarCov%*%(der)
SELnOdds <- sqrt(VarLnOdds)
LnOdds <- b0+b1*packsize
LCIlnOdds <- LnOdds - 1.96*SELnOdds
UCIlnOdds <- LnOdds + 1.96*SELnOdds
CIlnOdds[i,1] <- LCIlnOdds
CIlnOdds[i,2] <- LnOdds
CIlnOdds[i,3] <- UCIlnOdds
CIlnOdds[i,4] <- packsize
S <- exp(LnOdds)/(1+exp(LnOdds))
LCIS <- exp(LCIlnOdds)/(1+exp(LCIlnOdds))
UCIS <- exp(UCIlnOdds)/(1+exp(UCIlnOdds))
CIS[i,1] <- LCIS
CIS[i,2] <- S
CIS[i,3] <- UCIS
CIS[i,4] <- packsize
}

# Plot the log-odds for mortality
#par(mfrow=c(2, 2),oma=c(0, 1, 2, 1))
x <- CIlnOdds[,4]
y <- CIlnOdds[,1:3]
matplot(x,y,type="l",lty=2:1, col=1,
xlab = "Pack Size",
ylab = "Log-Odds of Mortality", cex.lab = 1.1,cex.axis = 1.3, font.lab = 2,
main = "ADULTS")


# Plot the estimated  mortality rate
x <- CIS[,4]
y <- CIS[,1:3]
matplot(x,y,type="l", lty=2:1, col=1,
xlab = "Pack Size",
ylab = "Annual Mortality Rate", cex.lab = 1.1, cex.axis = 1.3, font.lab = 2,
main = "ADULTS")

#add means for S, M, L packs with error bars in x and y

packsize <- c(4.8, 8.1, 15.1)
mortmean <- c(0.11, 0.17, 0.24)
mortupper95 <- c(0.11, 0.06, 0.065)
mortlower95 <- c(0.063, 0.05, 0.05)

packsizeSD <- c(0.69 ,1.56 ,2.8)
n<- c(46,185,163)
packsize95 <- (1.96*(packsizeSD/sqrt(n)))

pa
plotCI(packsize, mortmean, err="y", uiw = mortupper95,
liw = mortlower95, pch = 19, add=TRUE)
```

```
plotCI(packsize, mortmean, err="x", uiw = packsize95,
liw = packsize95, pch = 19, add=TRUE)
```