Lecture 9

GLMs for Binary Data
Statistics 211 - Statistical Methods II

Presented February 20, 2018
GLMs for binary outcomes

Consider the case of a binary outcome variable $Y$ which takes on the values 0 or 1

- Heart disease (yes/no)
- Voting result (Democrat/Republican)
- Faculty promotion (yes/no)

In this case, the random variable $Y$ follows a Bernoulli distribution with mean $\mu$ and variance $\mu(1 - \mu)$

\[
E[Y] = \Pr[Y = 1] = \mu \\
\text{Var}[Y] = \mu(1 - \mu)
\]
GLMs for binary outcomes

- Goal: Model the probability of a success as a function of some explanatory variable $X$ (only assume one covariate for now)

- Thus we will consider a model of the form:

$$g(\mu) = \beta_0 + \beta_1 X$$
GLMs for binary outcomes

**Specification of components of the GLM**

1. Systematic component (Done)
2. Random component (Done)
3. Link function (Need to decide)
‘Common’ link functions for binary data

Identity link function

- Linear (identity) link function
  - Identity link so that
    \[ \mu = \beta_0 + \beta_1 X \]
  - Interpretation: \( \beta_1 \) is the difference in the response probability comparing two populations differing by 1-unit in \( X \)
    - Modeling the risk difference (RD)
  - Potential Problem: Model assumes the outcome is unbounded even though we are modeling a probability (potential sacrifice of model fit over interpretability)
### Log link function

- **Log link function**

  - Log link so that

    \[ \log(\mu) = \beta_0 + \beta_1 X \]

  - **Interpretation**: \( e^{\beta_1} \) is the relative difference in the response probability comparing two populations differing by 1-unit in \( X \)

  - Modeling the *risk ratio* (RR)

- **Potential Problem**: Model assumes the outcome is unbounded even though we are modeling a log-probability with support between \(-\infty\) and 0 (potential sacrifice of model fit over interpretability)
‘Common’ link functions for binary data

Logit link function

- Logit link function
  - Logit link so that
    \[
    \text{logit}(\mu) = \log\left(\frac{\mu}{1 - \mu}\right) = \beta_0 + \beta_1 X
    \]
  - This is the simple logistic regression model
  - Interpretation: \(e^{\beta_1}\) is the relative difference in the odds of ‘success’ comparing two populations differing by 1-unit in \(X\)
    - Modeling the odds ratio (OR)
  - Nice property: The log-odds has support between \(-\infty\) and \(\infty\)
‘Common’ link functions for binary data

Logit link function

- Probability response curve as a function of $X$ for the logit model
  - Under the simple logistic model, $\mu = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$
  - Note: The function $f(x) = e^x/(1 + e^x)$ is called the expit

- $\beta_1 > 0$ implies that the probability of a ‘success’ increases with $X$
- $\beta_1 < 0$ implies that the probability of a ‘success’ decreases with $X$
‘Common’ link functions for binary data

Logit link function

▶ Example: \( \text{logit}(\mu) = \log \left( \frac{\mu}{1-\mu} \right) = 1 + 0.2X \)
‘Common’ link functions for binary data

Logit link function

Example: \( \text{logit}(\mu) = \log \left( \frac{\mu}{1-\mu} \right) = 1 - 0.2X \)
‘Common’ link functions for binary data

Probit link function

Probit link function

Recall that the cumulative distribution function (CDF) of a random variable $X$ is given by

$$F(x) = \Pr[X \leq x]$$

The S-shaped probability response curve ($\beta_1 > 0$) for the logistic model corresponds to the CDF for the logistic distribution.

This motivates the use of another class of link functions by taking $\mu(x) = F(x)$ for some CDF.
‘Common’ link functions for binary data

Probit link function

- The most popular choice of $F$ is that corresponding to the standard normal distribution

- Denote the CDF corresponding to the standard normal distribution as $\Phi(\cdot)$ so that

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_0^z e^{\frac{1}{2}x^2} dx$$
‘Common’ link functions for binary data

Probit link function

Then we can consider a model of the mean given by

$$\mu = \Phi(\beta_0 + \beta_1 X)$$

or equivalently,

$$\Phi^{-1}(\mu) = \beta_0 + \beta_1 X$$

The link function $\Phi^{-1}(\cdot)$ is called the *probit* link
‘Common’ link functions for binary data

Comparison of fitted response probabilities

- Over mid-range values of the linear predictor \( z = \beta_0 + \beta_1 X \)
  (or \( \mu \)), the linear, probit, and logit models agree

- This is because

\[
\text{expit}(z) \approx \Phi \left( \frac{\sqrt{2\pi}z}{4} \right) \approx \frac{1}{2} + \frac{z}{4}, \quad \text{for } -2 \leq z \leq 2
\]

- The main reason that logits are often preferred to probits is because coefficients from the logistic model are interpretable (odds ratios)

- If prediction of probabilities is the focus, then either model can be considered
‘Common’ link functions for binary data

Logit link function

- Compare the response curves for each of the (appropriately scaled) linear predictors:
Example - Modeling the probability of CHD in the Framingham Study

Background on the Framingham study

- 5209 subjects identified in 1948 in a small Massachusetts town
- Biennial exams for blood pressure, serum cholesterol, and relative weight
- 30 year followup data available from course website
- Major endpoints include the occurrence of coronary heart disease (CHD) and deaths from
  - CHD or MI
  - Cerebrovascular accident (CVA or stroke)
  - Cancer
  - Other causes

Scientific goal

- Quantify the prevalence of CHD at the followup exam among males age 30+
Example - Modeling the probability of CHD in the Framingham Study

> framingham <- read.table("http://www.ics.uci.edu/~dgillen/STAT211/Data/Framingham.txt", header=TRUE)

> framingham[1:5,]

<table>
<thead>
<tr>
<th>sex</th>
<th>sbp</th>
<th>dbp</th>
<th>scl</th>
<th>chdfate</th>
<th>followup</th>
<th>age</th>
<th>bmi</th>
<th>month</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1120</td>
<td>80</td>
<td>267</td>
<td>1</td>
<td>18</td>
<td>55</td>
<td>25.0</td>
<td>8</td>
<td>2642</td>
</tr>
<tr>
<td>2</td>
<td>130</td>
<td>78</td>
<td>192</td>
<td>1</td>
<td>35</td>
<td>53</td>
<td>28.4</td>
<td>12</td>
<td>4627</td>
</tr>
<tr>
<td>3</td>
<td>144</td>
<td>90</td>
<td>207</td>
<td>1</td>
<td>109</td>
<td>61</td>
<td>25.1</td>
<td>8</td>
<td>2568</td>
</tr>
<tr>
<td>4</td>
<td>92</td>
<td>66</td>
<td>231</td>
<td>1</td>
<td>147</td>
<td>48</td>
<td>26.2</td>
<td>11</td>
<td>4192</td>
</tr>
<tr>
<td>5</td>
<td>162</td>
<td>98</td>
<td>271</td>
<td>1</td>
<td>169</td>
<td>39</td>
<td>28.4</td>
<td>11</td>
<td>3977</td>
</tr>
</tbody>
</table>

> summary(framingham)

> # Recode sex to something obvious (sex=1 -> female)
> framingham$sex <- framingham$sex - 1
> names(framingham)[1] <- "female"
Fitting GLMs in R is done with the `glm` function

```r
> help( glm )
```

**Description**

`glm()` is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

**Usage**

```r
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart,
    offset, control = glm.control(...), model = TRUE,
    method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)
```

**Arguments**

- **formula**: a symbolic description of the model to be fit. The details of model specification are given below.
- **family**: a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
- **data**: an optional data frame containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `glm` is called.
- **weights**: an optional vector of weights to be used in the fitting process.
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **na.action**: a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The "factory-fresh" default is `na.omit`.

Binomial Regression
Logistic model to estimate the association between SBP and the odds of CHD

```R
> ##
> ###### Logistic model
> ##
> ##
> fit.logit <- glm( chdfate ~ sbp, data=framingham, family=binomial(link="logit") )
> fit.logit

Call: glm(formula = chdfate ~ sbp, family = binomial(link = "logit"),
          data = framingham)

Coefficients:
(Intercept) sbp
     -3.00881  0.01659

Degrees of Freedom: 4698 Total (i.e. Null); 4697 Residual
Null Deviance: 5844 Residual Deviance: 5696 AIC: 5700
```
Logistic model to estimate the association between SBP and the odds of CHD

> summary( fit.logit )

Call:
glm(formula = chdfate ~ sbp, family = binomial(link = "logit"),
    data = framingham)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8320  -0.8668  -0.7634   1.3676   1.8368

Coefficients:
             Estimate Std. Error  z value Pr(>|z|)
(Intercept) -3.00881   0.18982  -15.850  < 2e-16 ***
sbp          0.01659   0.00138   11.980  < 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: 5844.1  on 4698 degrees of freedom
Residual deviance: 5695.7  on 4697 degrees of freedom
AIC: 5699.7

Number of Fisher Scoring iterations: 4

Interpretation:
Logistic model to estimate the association between SBP and the odds of CHD

```
> ##
> #### Refit the model, scaling sbp per 10 mmHg
> ##
> fit.logit <- glm( chdfate ~ I(sbp/10), data=framingham,
>                   family=binomial(link="logit") )
```

```
> ##
> #### Use glmCI() function on course webpage to exponential coefficients and form CI’s
> ##
> glmCI( fit.logistic )

|             | exp(Est) | ci95.lo | ci95.hi | z.value | Pr(>|z|) |
|-------------|----------|---------|---------|---------|----------|
| (Intercept) | 0.04935042 | 0.03401885 | 0.0715916 | -15.85123 | 1.378551e-56 |
| I(sbp/10)   | 1.18049490 | 1.14887319 | 1.2129870 | 11.97785 | 4.642017e-33 |
```

Interpretation: The odds of CHD are estimated to be 18.1% higher when comparing two populations, one of which has systolic blood pressure 10 mmHg higher than the other (95% CI: 14.9%, 21.3%).
Probit model to estimate the association between SBP and the odds of CHD

```r
> ##
> ###### Probit model
> ##
> > fit.probit <- glm( chdfate ~ I(sbp/10), data=framingham, 
> family=binomial(link="probit") )
> summary( fit.probit )

Call:
glm(formula = chdfate ~ I(sbp/10), family = binomial(link = "probit"),
    data = framingham)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-1.8426  -0.8680  -0.7618   1.3660   1.8506

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)     
(Intercept)  -1.852951    0.114407  -16.20   <2e-16 ***
I(sbp/10)     0.102093     0.008407   12.14   <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: 5844.1  on 4698  degrees of freedom
  Residual deviance: 5694.3  on 4697  degrees of freedom
AIC: 5698.3

Number of Fisher Scoring iterations: 4
```
Example - Modeling the probability of CHD in the Framingham Study

Comparison of logit and probit models

- Interpretation of the probit model:
  - Assumes each individual has a latent continuous measure of CHD that follows a standard normal distribution
  - Slope coefficient in the probit model is the expected difference in this latent measure (a standard normal quantile) comparing two populations differing by 10mmHg in SBP
  - Hmmm... Maybe the fitted probabilities differ between the two models.
Comparison of fitted probabilities from the two models

> #
> ###### Probit model
> #
> fit.probit <- glm( chdfate ~ I(sbp/10), data=framingham, 
>                   family=binomial(link="probit") )

> summary( fit.probit )

> plot( sort( framingham$sbp ), sort( fitted( fit.logit ) ),
>       type="l", col="red", ylab="Estimated probability of CHD",
>       xlab="Systolic BP (mmHg)" )
> lines( sort( framingham$sbp ), sort( fitted( fit.probit ) ), lty=2 )
> table( sort( framingham$sbp ), sort( fitted( fit.logit ) ) )

<table>
<thead>
<tr>
<th>(80,114]</th>
<th>(114,124]</th>
<th>(124,134]</th>
<th>(134,148]</th>
<th>(148,270]</th>
</tr>
</thead>
<tbody>
<tr>
<td>976</td>
<td>964</td>
<td>931</td>
<td>900</td>
<td>925</td>
</tr>
</tbody>
</table>

> sbpgrp <- cut( framingham$sbp, quantile( framingham$sbp, seq(0,1,.2) ) )
> empirical.p <- table( framingham$chdfate, sbpgrp )[2,] / table( sbpgrp )
> points( unlist( lapply( split( framingham$sbp, sbpgrp ), mean ) ), empirical.p )
Comparison of fitted probabilities from the two models

![Graph showing comparison of fitted probabilities from logit and probit models](image-url)
Comparison of coefficients from the two models

**Comparison of coefficients**

- The coefficients should roughly match up if we ‘standardize’ them

```r
> logitbeta <- fit.logit$coef
> logitbeta

(Intercept)    I(sbp/10)
-3.0088090   0.1659338

> probitbeta <- fit.probit$coef
> probitbeta

(Intercept)    I(sbp/10)
-1.8529512   0.1020929

> probitbeta / (sqrt(2*pi)/4)

(Intercept)    I(sbp/10)
-2.9568663   0.1629168
```

**Note:** The coefficients would not be so close if the probability of CHD were near 0 or 1
Binary vs. Binomial Regression

Binomial Data

- If data are inherently grouped (all categorical predictors) then it can be advantageous to store and analyze the data in a *collapsed* form
  - More efficient use of memory
  - Better for performing goodness-of-fit tests (later)
Binary vs. Binomial Regression

Example: Framingham Data

- Suppose that we were only interested in categorical exposure covariates:

```r
> ##
> ###### Comparison of Binary vs Binomial Regression
> ##
> sbpgrp <- cut( framingham$sbp, c(0,100,125,150,175,200,225,275) )
> agegrp <- cut( framingham$age, c(0,40,50,70) )
> bmigrp <- cut( framingham$bmi, c(0,20,25,30,60) )
> framgrp <- as.data.frame( cbind( framingham$female, agegrp, bmigrp,
>                                 sbpgrp, framingham$chdfate ) )
> names( framgrp ) <- c("female", "agegrp", "bmigrp", "sbpgrp", "chdfate" )

> framgrp[1:5,]
female agegrp bmigrp sbpgrp chdfate
1 0 3 2 2 1
2 0 3 3 3 1
3 0 3 3 3 1
4 0 2 3 1 1
5 0 1 3 4 1
```
Binary vs. Binomial Regression

Fit binary regression using `glm()`

- The dataset now contains a total of 4690 observations (1 record per individual)

- One possibility is to keep the data in this fashion and analyze each individual separately representing a Bernoulli outcome (CHD: yes/no)

```r
> dim( framgrp )
[1] 4690 5

> fit.binary <- glm( chdfate ~ female + factor(agegrp) + factor(bmigrp) + factor(sbpgrp), data=framgrp, family=binomial )

> glmCI( fit.binary )

|            | exp( Est ) | ci95.lo | ci95.hi | z value | Pr(>|z|) |
|------------|------------|---------|---------|---------|----------|
| (Intercept)| 0.1004836  | 0.05330377 | 0.1894227 | -7.103503 | 1.216340e-12 |
| female     | 0.4765312  | 0.41754381 | 0.5438518 | -10.993856 | 4.090664e-28 |
| factor(agegrp)2 | 1.3746793 | 1.16519485 | 1.6218258 | 3.772410 | 1.616780e-04 |
| factor(agegrp)3 | 1.6830038 | 1.41697893 | 1.9989724 | 5.930254 | 3.024668e-09 |
| factor(bmigrp)2 | 1.8568520 | 1.27238784 | 2.7097865 | 3.209070 | 1.331651e-03 |
| factor(bmigrp)3 | 2.3725291 | 1.62290512 | 3.4684064 | 4.459181 | 8.227335e-06 |
| factor(bmigrp)4 | 2.9383140 | 1.95860473 | 4.4080815 | 5.208335 | 1.905422e-07 |
| factor(sbpgrp)2 | 1.8327248 | 1.06314550 | 3.1593797 | 2.180343 | 2.923201e-02 |
| factor(sbpgrp)3 | 2.5992414 | 1.50735752 | 4.4820527 | 3.436095 | 5.901632e-04 |
| factor(sbpgrp)4 | 3.1702936 | 1.80037476 | 5.5825942 | 3.996706 | 6.422986e-05 |
| factor(sbpgrp)5 | 3.4503162 | 1.85585038 | 6.4146776 | 3.914303 | 9.066589e-05 |
| factor(sbpgrp)6 | 7.3262683 | 3.32039231 | 16.1650199 | 4.932126 | 8.133952e-07 |
| factor(sbpgrp)7 | 11.9961057 | 3.22609969 | 44.606764 | 3.707958 | 2.089374e-04 |
```
Binary vs. Binomial Regression

Collapse the data for binomial regression

- Now, collapse the data, removing repeated patterns of covariate values
  - Keep track of the frequency of each combination of chdfate, sbpgrp, agegrp, bmigrp, and female values

```r
> collapse <- function( data, outcome ){
+   index <- (1:length(names(data)))[ names(data)==outcome ]
+   y <- data[,index]
+   data <- data[-index]
+   rslt <- aggregate( y, data, FUN=length)
+   rslt <- as.data.frame( cbind( rslt, aggregate(y, data, FUN=sum)[dim(rslt)[2]] )
+   names( rslt ) <- c( names(data), "n", paste("n.", outcome, sep=""))
+   rslt}
>
> framgrp <- collapse( framgrp, "chdfate" )
> dim( framgrp )
[1] 129 6

> framgrp[1:10,]
  female agegrp bmigrp sbpgrp n n.chdfate
1   0     1     1     1  1  1
2   1     1     1     1 15  1
3   0     2     1     1  4  0
4   1     2     1     1  6  0
5   0     3     1     1  1  0
6   1     3     1     1  3  0
7   0     1     2     1  8  1
8   1     1     2     1 37  1
9   0     2     2     1  7  1
10  1     2     2     1 16  1
```
Binary vs. Binomial Regression

Collapse the data for binomial regression

- Now we can use the (frequency) weights options in glm to analyze the data

- The effect of using frequency weights is the same as expanding the dataset, creating identical records whose multiplicity is specified by weights

- Expanding only takes place at analysis time, behind the scenes

```r
> fit.binom <- glm( n.chdfate/n ~ female + factor( agegrp ) +
                   factor( bmgirp ) + factor( sbpgrp ), data=framgrp,
                   weights=n, family=binomial )

> glmCI( fit.binom )

                      exp( Est )  ci95.lo  ci95.hi  z value  Pr(>|z|)
(Intercept)         0.1004836 0.05330196 0.1894291 -7.103124 1.219676e-12
female              0.4765312 0.41754372 0.5438519 -10.993840 4.091388e-28
factor(agegrp)2     1.3746793 1.16519446 1.6218264  3.772403 1.616830e-04
factor(agegrp)3     1.6830038 1.41697854 1.9989729  5.930244 3.024845e-09
factor(bmgirp)2     1.8568520 1.27237820 2.7098071  3.209006 1.331949e-03
factor(bmgirp)3     2.3725291 1.62289310 3.4684321  4.459094 8.230675e-06
factor(bmgirp)4     2.9383140 1.95859114 4.4081121  4.408112 5.208246e-07
factor(sbpgrp)2     1.8327249 1.06310984 3.1594858  2.180209 2.924196e-02
factor(sbpgrp)3     2.5992144 1.50730715 4.4822026  3.435885 5.906226e-04
factor(sbpgrp)4     3.1702937 1.80031685 5.5827739  3.996479 6.429152e-05
factor(sbpgrp)5     3.4503163 1.85579585 6.4148663  3.914117 9.073561e-05
factor(sbpgrp)6     7.3262684 3.32031596 16.1653920  4.931982 8.139924e-07
factor(sbpgrp)7     11.9961059 3.22605069 44.6076551  4.091388 2.089728e-04
```