Chapter 5: Generalized Linear Modeling

Tyson S. Barrett

Summer 2017

Utah State University

Introduction

Logistic Regression

Poisson Regression

Beta Regression

Conclusions

Introduction

Good Quote

"You must stick to your conviction, but be ready to abandon your assumptions."
— Dennis Waitley

GLMs

Generalized Linear Models (GLMs):

1. Are extensions of linear regression to areas where assumptions of normality and homoskedasticity do not hold
2. There are several versions of GLM's, each for different types and distributions of outcomes.

We are going to go through several of the most common GLMs.

Types

We discuss:

1. Logistic Regression
2. Poisson Regression
3. GLM with Gamma distribution
4. Negative binomial
5. Beta Regression

Logistic Regression

Logistic Regression

For binary outcomes (e.g., yes or no, correct or incorrect, sick or healthy)

```
## First creating binary depression variable
df <- df %>%
  mutate(dep = dpq010 + dpq020 + dpq030 + dpq040 + dpq050 +
                dpq060 + dpq070 + dpq080 + dpq090) %>%
  mutate(dep2 = ifelse(dep >= 10, 1,
                 ifelse(dep < 10, 0, NA)))
```

Note:

1. IF depression ≥ 10 then dep2 is 1,
2. IF dpression < 10, then dep2 is 0,
3. ELSE dep2 is NA.

Running Logistic Regression

Since the outcome is binary, we use a statistical transformation to make things work well. This makes it so the outcome is in "log-odds." A simple exponentiation of the coefficients and we get very useful "odds ratios."

Luckily, running a logistic regression is simple in `R`. We first create the binary outcome variable called `dep`. We use a new function called `mutate` to create a new variable (we could do this a number of ways but this is probably the cleanest way).

Running Logistic Regression

```
## Fix some placeholders
df <- df %>%
  mutate(asthma = washer(mcq010, 9),
         asthma = washer(asthma, 2, value = 0)) %>%
  mutate(sed = washer(pad680, 9999, 7777))
```

Running Logistic Regression

Now let's run the logistic regression:

```
l_fit <- glm(dep2 ~ asthma + sed + race + famsize,
             data = df,
             family = "binomial")
summary(l_fit)
```

```
##
## Call:
## glm(formula = dep2 ~ asthma + sed + race + famsize, family = "binomi
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7831  -0.4479  -0.4078  -0.3645   2.5471
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.6203555  0.2380770 -11.006  < 2e-16 ***
```

Output of Logistic Regression

We used `glm()` (stands for generalized linear model)

The key to making it logistic, since you can use `glm()` for a linear model using maximum likelihood instead of `lm()` with least squares, is `family = "binomial"`

Poisson Regression

Poisson Regression

Again, we will use the `glm()` function.

The difference here is we will be using an outcome that is a count variable. For example, the sedentary variable (`sed`) that we have in `df` is a count of the minutes of sedentary activity.

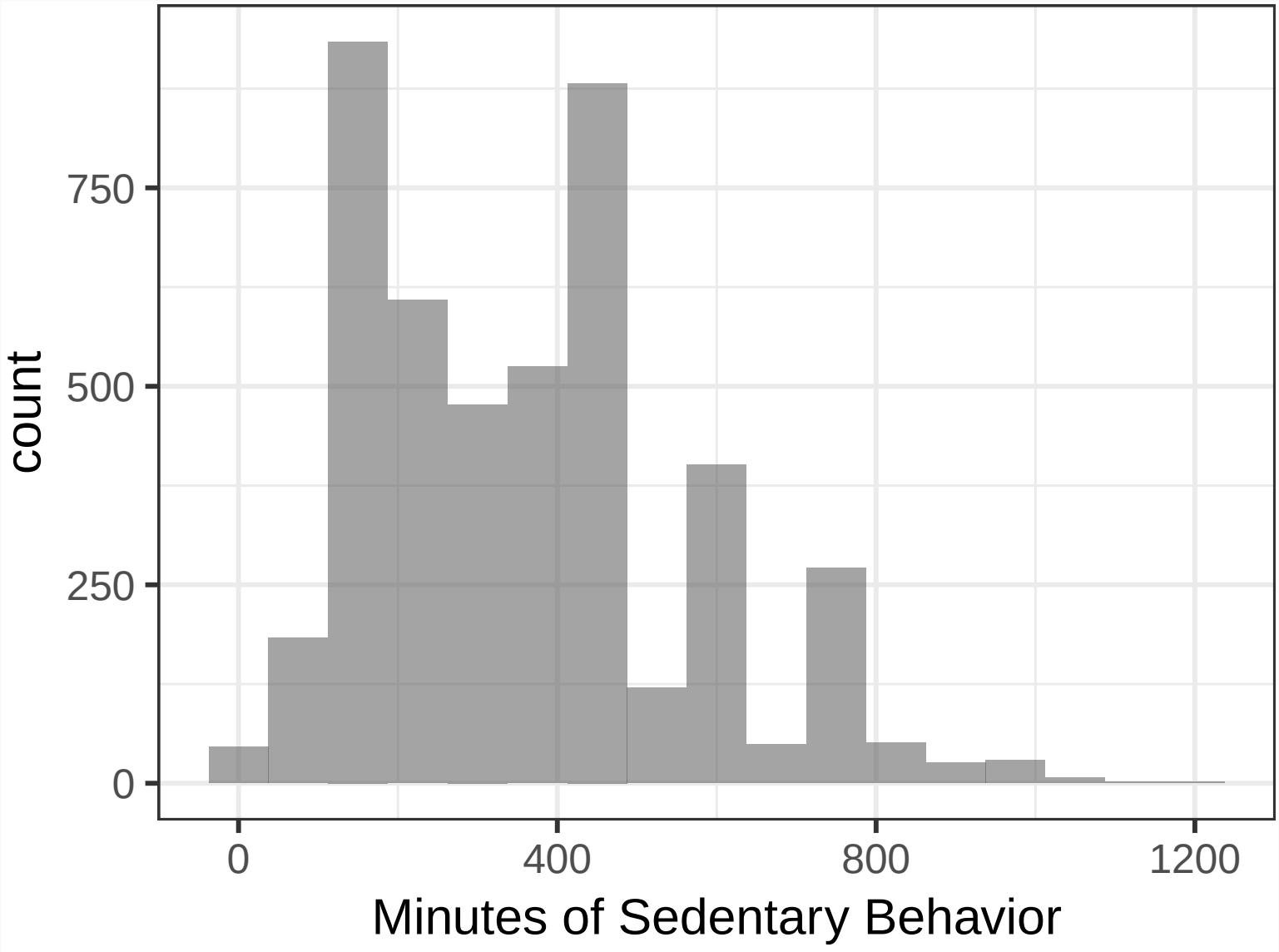
Running Poisson Regression

```r
p_fit <- glm(sed ~ asthma + race + famsize,
             data = df,
             family = "poisson")
summary(p_fit)
```

```
##
## Call:
## glm(formula = sed ~ asthma + race + famsize, family = "poisson",
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -27.362   -8.430   -1.477    5.823   34.507
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        5.6499871  0.0035550 1589.31   <2e-16 ***
## asthma             0.0614965  0.0021434   28.69   <2e-16 ***
## raceOtherHispanic  0.1393438  0.0040940   34.04   <2e-16 ***
## raceWhite          0.3484622  0.0033438  104.21   <2e-16 ***
## raceBlack          0.3400346  0.0034430   98.76   <2e-16 ***
## raceOther          0.3557953  0.0036273   98.09   <2e-16 ***
## famsize           -0.0188673  0.0005488  -34.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 496351  on 4436  degrees of freedom
## Residual deviance: 475428  on 4430  degrees of freedom
##   (195 observations deleted due to missingness)
## AIC: 508999
##
## Number of Fisher Scoring iterations: 5
```

Running Poisson Regression

Sedentary may be over-dispersed:



and so other methods related to poisson may be necessary. For this book, we are

Gamma

- very similar to poisson but does not require integers and can handle more dispersion.
- the outcome must have values > 0.

```
## Adjust sed
df$sed_gamma <- df$sed + .01
g_fit <- glm(sed_gamma ~ asthma + race + famsize,
             data = df,
             family = "Gamma")
summary(g_fit)
```

```
##
## Call:
## glm(formula = sed_gamma ~ asthma + race + famsize, family = "Gamma",
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3589  -0.4613  -0.0845   0.2926   1.6868
```

Two-Part or Hurdle Models

We are going to use the `pscl` package to run a hurdle model. These models are built for situations where there is a count variable with many zeros ("zero-inflated"). The hurdle model makes slightly different assumptions regarding the zeros than the pure negative binomial that we present next. The hurdle consists of two models: one for whether the person had a zero or more (binomial) and if more than zero, how many (poisson).

To run a hurdle model, we are going to make a sedentary variable with many more zeros to illustrate and then we will run a hurdle model.

```
## Zero inflated sedentary (don't worry too much about the specifics)
df$sed_zero <- ifelse(sample(1:100,
                             size = length(df$sed),
                             replace=TRUE) %in% c(5,10,11,20:25), 0,
                      df$sed)
## Hurdle model
library(pscl)
h_fit = hurdle(sed_zero ~ asthma + race + famsize,
```

Negative Binomial

Similar to that above, negative binomial is for zero-inflated count variables. It makes slightly different assumptions than the hurdle and doesn't use a two-part approach. In order to run a negative binomial model we'll use the `MASS` package and the `glm.nb()` function.

```
library(MASS)
fit_nb <- glm.nb(sed_zero ~ asthma + race + famsize,
                 data = df)
summary(fit_nb)
```

Note that this model is not really appropriate because our data is somewhat contrived.

Beta Regression

Beta Regression

For outcomes that are bound between a lower and upper bound, Beta Regression is a great method. For example, if we are looking at test scores that are bound between 0 and 100. It is a very flexible method and allows for some extra analysis regarding the variation.

Running Beta Regression

For this, we are going to use the betareg package. But first, we are going to reach a little and create a ficticiously bound variable in the data set.

```
## Variable bound between 0 and 1
df$beta_var <- sample(seq(.05, .99, by = .01),
                      size = length(df$asthma),
                      replace = TRUE)
library(betareg)
fit_beta <- betareg(beta_var ~ asthma + race + famsize,
                    data = df)
summary(fit_beta)
```

Running Beta Regression

```
##
## Call:
## betareg(formula = beta_var ~ asthma + race + famsize, data =
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -2.0937 -0.7133 -0.0425  0.6083  2.9121
##
## Coefficients (mean model with logit link):
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.183438   0.063182   2.903  0.00369 **
## asthma             0.079145   0.043669   1.812  0.06993 .
## raceOtherHispanic -0.028195   0.071914  -0.392  0.69501
## raceWhite         -0.052655   0.058669  -0.897  0.36946
## raceBlack         -0.013716   0.060909  -0.225  0.82184
## raceOther          0.005879   0.065285   0.090  0.92825
## famsize           -0.017376   0.010808  -1.608  0.10789
```

Conclusions

Conclusions

There are many resources available to learn more about beta regression and each of these GLM's. As for now, we are going to move on to more complex modeling where there are clustering or repeated measures in the data.

Conclusions

One of the great things about `R` is that most modeling is very similar to the basic `lm()` function. In all of these GLM's the arguments are nearly all the same: a formula, the data, and family of model. As you'll see for Multilevel and Other Models chapters, this does not change much. Having a good start with basic models and GLM's gets you ready for nearly every other modeling type in `R`.