Stat 5421 Lecture Notes: To Accompany Agresti Ch 8

Charles J. Geyer

July 27, 2023

Multinomial Response

Response (in Scare Quotes)

To say we are treating one dimension of a contingency table as a "response" is the same as saying we want a product multinomial model with all of the multinomials being for the "response" variable. This means two things.

- In order for product multinomial and Poisson to be the same, we must have the interaction of all other factors in the model formula.
- In order to have any other term on the right-hand side of the formula, it must be "interacted with" the "response" variable.

The latter is because we want to know about the effect of other terms on the "response".

Using R Function Loglm

We do the example on alligator food choice (Section 8.1.2 in Agresti)

```
library(CatDataAnalysis)
data(table_8.1)
names(table_8.1)
```

```
## [1] "lake"   "gender" "size"   "food"   "count"
```

```
sapply(table_8.1, class)
```

```
##      lake    gender      size      food     count
## "integer" "integer" "integer" "integer" "integer"
```

```
lapply(table_8.1, unique)
```

```
## $lake
## [1] 1 2 3 4
##
## $gender
## [1] 1 2
##
## $size
## [1] 1 2
##
## $food
## [1] 1 2 3 4 5
##
## $count
##  [1]  7  1  0  5  4  2 16  3 13  6  9  8 10
```

These data are given in really inconvenient coding. Let's recode using the variable names in Table 8.1, which we hope are given in the correct order. (We'll check that.)

```
foo <- transform(table_8.1,
    lake = factor(lake,
        labels = c("Hancock", "Oklawaha", "Trafford", "George")),
    gender = factor(gender, labels = c("Male", "Female")),
    size = factor(size, labels = c("<=2.3", ">2.3")),
    food = factor(food,
        labels = c("Fish", "Invertebrate", "Reptile", "Bird", "Other")))
bar <- xtabs(count ~ size + food + gender + lake, data = foo)
bar
```

```
## , , gender = Male, lake = Hancock
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3    7            1       0    0     5
##   >2.3     4            0       0    1     2
##
## , , gender = Female, lake = Hancock
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3   16            3       2    2     3
##   >2.3     3            0       1    2     3
##
## , , gender = Male, lake = Oklawaha
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3    2            2       0    0     1
##   >2.3    13            7       6    0     0
##
## , , gender = Female, lake = Oklawaha
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3    3            9       1    0     2
##   >2.3     0            1       0    1     0
##
## , , gender = Male, lake = Trafford
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3    3            7       1    0     1
##   >2.3     8            6       6    3     5
##
## , , gender = Female, lake = Trafford
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3    2            4       1    1     4
##   >2.3     0            1       0    0     0
##
## , , gender = Male, lake = George
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3   13           10       0    2     2
##   >2.3     9            0       0    1     2
##
## , , gender = Female, lake = George
##
##        food
## size    Fish Invertebrate Reptile Bird Other
##   <=2.3    3            9       1    0     1
##   >2.3     8            1       0    0     1
```

That matches Table 8.1 in Agresti. Note that we had to write the formula `count ~ size + food + gender + lake` for R function `xtabs` with the terms in that order to get our printout to match the table in Agresti.

Now we want to consider `food` to be the "response" variable. So we fit all of the models in the first three columns of Table 8.2 in Agresti using R function `loglm` in R package `MASS` because it gives both Pearson Chi-Square and likelihood ratio test statistics. (We could also use R function `glm` but it does not give the Pearson statistics.)

```
names(dimnames(bar))
```

```
## [1] "size"   "food"   "gender" "lake"
```

```
library(MASS)
lout <- list()
lout[["empty"]] <- loglm(~ size * gender * lake + food, bar)
lout[["gender"]] <- loglm(~ size * gender * lake + food * gender, bar)
lout[["size"]] <- loglm(~ size * gender * lake + food * size, bar)
lout[["lake"]] <- loglm(~ size * gender * lake + food * lake, bar)
lout[["size.lake"]] <- loglm(~ size * gender * lake + food * (size + lake), bar)
lout[["size.gender.lake"]] <- loglm(~ size * gender * lake + food * (size + gender + lake), bar)
woof <- lapply(lout, function(x) c(lrt = x$lrt, pearson = x$pearson, df = x$df))
woof <- as.data.frame(woof)
woof <- as.matrix(woof)
woof <- t(woof)
round(woof, 1)
```

```
##                    lrt pearson df
## empty            116.8   106.5 60
## gender           114.7   101.2 56
## size             101.6    86.9 56
## lake              73.6    79.6 48
## size.lake         52.5    58.0 44
## size.gender.lake  50.3    52.5 40
```

This agrees with the left half of Table 8.2 in Agresti. Agresti says LRT is more reliable than Pearson, but since the sample size is not "large" with many small cell counts, neither is accurate. If we really wanted accurate P-values, we would need to parametric bootstrap, as discussed in the notes on Chapter 9.

Using R Function Glmbb

R function `glmbb` in R package `glmbb` fits models using R function `glm`. It considers all models in a range of models. There will be a handout for that. But for now, we just show it without much explanation.

```
library(glmbb)
bout <- glmbb(big = count ~ lake * gender * size * food,
    little = ~ lake * gender * size + food,
    family = poisson, data = foo)
```

```
## Warning: glm.fit: fitted rates numerically 0 occurred

## Warning: glm.fit: fitted rates numerically 0 occurred
```

```
summary(bout)
```

```
##
## Results of search for hierarchical models with lowest AIC.
## Search was for all models with AIC no larger than min(AIC) + 10
## These are shown below.
##
##   criterion  weight    formula
##   288.0      0.903304  count ~ lake*food + size*food + lake*gender*size
##   293.7      0.050072  count ~ lake*food + gender*food + size*food + lake*gender*size
##   294.9      0.028388  count ~ lake*gender*size + lake*size*food
##   296.8      0.010643  count ~ size*food + lake*gender*size + lake*gender*food
##   297.5      0.007593  count ~ gender*food + lake*gender*size + lake*size*food
```

This says that by far the best model is the one Agresti calls $L+S$. As we have seen, the formula for this model, considered as a GLM or as a log-linear model is `count ~ lake*food + size*food + lake*gender*size`.

These models having AIC within 10 of the minimum AIC are in Agresti's notation

$$L + S$$
$$G + L + S$$
$$L * S$$
$$G * L + S$$
$$G + L * S$$

In the notation of R function `glmbb` these are the models between

```
bout$little
```

```
## count ~ lake * gender * size + food
## <environment: 0x55970e9146d8>
```

```
bout$big
```

```
## count ~ lake * gender * size * food
```

(inclusive) that have AIC no larger than min(AIC) + 10.

Ordered Categorical Data

This topic is Section 8.2 in Agresti.

We are going to use R function `polr` in R package `MASS` to do what it calls *proportional odds logistic regression* (POLR). Because its definitions may differ from Agresti's, we will take its help as authoritative. In the "details" section of the help for this function, it says

> This model is what Agresti [they cite the second edition of our textbook, whereas we are using the third edition] calls a *cumulative link* model. The basic interpretation is as a *coarsened* version of a latent variable Y_i which has a logistic or normal or extreme-value or Cauchy distribution with scale parameter one and a linear model for the mean. The ordered factor which is observed is which bin Y_i falls into with breakpoints
>
> $$\zeta_0 = -\infty < \zeta_1 < \cdots < \zeta_K = \infty$$
>
> This leads to the model
>
> $$\operatorname{logit} P(Y \leq k | x) = \zeta_k - \eta$$
>
> with *logit* replaced by *probit* for a normal latent variable, and η being the linear predictor, a linear function of the explanatory variables (with no intercept). Note that it is quite common for other software to use the opposite sign for η (and hence the coefficients `beta`).
>
> In the logistic case, the left-hand side of the last display is the log odds of category k or less, and since these are log odds which differ only by a constant for different k, the odds are proportional. Hence the term *proportional odds logistic regression*.

Comparing with equation (8.5) in the third edition of Agresti (our textbook) we see that R function `polr` does indeed use the opposite sign for regression coefficients from what Agresti uses (and apparently also from what the software he uses for his examples uses). So we had better stick with what R does.

But we also see that other than a possible change of sign of regression coefficients, R function `polr` agrees with the model discussed in Section 8.2 of Agresti.

As an example, we do example 8.2.4 in Agresti.

```
# clean up R global environment
rm(list = ls())

data(table_8.5)
names(table_8.5)
```

```
## [1] "race"   "trauma" "happy"
```

```
sapply(table_8.5, class)
```

```
##      race    trauma     happy
## "integer" "integer" "integer"
```

```
lapply(table_8.5, unique)
```

```
## $race
## [1] 0 1
##
## $trauma
## [1] 0 1 2 3 4 5
##
## $happy
## [1] 1 2 3
```

It is annoying that Agresti has coded these data numerically when they should be categorical. Let's fix that like we did for the alligator data.

```
foo <- transform(table_8.5,
    race = factor(race, labels = c("White", "Black")),
    happy = ordered(happy,
        labels = c("Very.happy", "Pretty.happy", "Not too happy")))
```

Now we are ready to invoke R function `polr`

```r
library(MASS)
pout <- polr(happy ~ race + trauma, data = foo)
summary(pout)
```

```
##
## Re-fitting to get Hessian

## Call:
## polr(formula = happy ~ race + trauma, data = foo)
##
## Coefficients:
##            Value Std. Error t value
## raceBlack 2.0361     0.6859   2.968
## trauma    0.4056     0.1830   2.216
##
## Intercepts:
##                              Value   Std. Error t value
## Very.happy|Pretty.happy    -0.5181  0.3400    -1.5238
## Pretty.happy|Not too happy  3.4006  0.5680     5.9872
##
## Residual Deviance: 148.407
## AIC: 156.407
```

We see that the "coefficients" agree with those in the SAS output shown in Agresti Table 8.6 except for sign, which may be due to R dropping a different dummy variable for `race` than SAS does or due to R using a different sign convention (discussed in the quote from the help page above). So we must be fitting the same model.

If we want to be sure what the signs of the coefficients mean, we can look at predictions.

```r
predict(pout, type = "probs",
    newdata = data.frame(race = c("White", "Black"),
    trauma = rep(2, 2)))
```

```
##   Very.happy Pretty.happy Not too happy
## 1 0.20928237    0.7208972    0.06982045
## 2 0.03339528    0.6015130    0.36509170
```

```r
predict(pout, type = "probs",
    newdata = data.frame(race = rep("White", 6), trauma = 0:5))
```

```
##   Very.happy Pretty.happy Not too happy
## 1 0.37329388    0.5944293    0.03227686
## 2 0.28420782    0.6681410    0.04765122
## 3 0.20928237    0.7208972    0.06982045
## 4 0.14997095    0.7488215    0.10120759
## 5 0.10523160    0.7502560    0.14451244
## 6 0.07269745    0.7251240    0.20217855
```

One says keeping `trauma` fixed at 2, there are more `Very happy` whites and more `Not too happy` blacks. And the other says keeping `race` fixed at `"White"`, more `trauma` makes for fewer `Very happy` and more `Not too happy`.

So in both cases we have positive "coefficient" means worse outcome in a problem like this were later in the order of the ordered categorical response means worse.