MLE - Lab 9

Andy Ballard

March 10, 2017

First, let's set up our workspace

Today

- Homework 4 (NOT due during break. Breaks are breaks.)
- Count models

Count models

When do we use count models?

When we want to model some sort of count variable, where the dependent variable has bounds $[0, \infty]$. Because we assume there is no upper limit, the support of the function goes from 0 to infinity. The Poisson distribution is a common underlying distribution for count models, and we'll use a Poisson model first. The Poisson distribution is related to a number of other distributions:

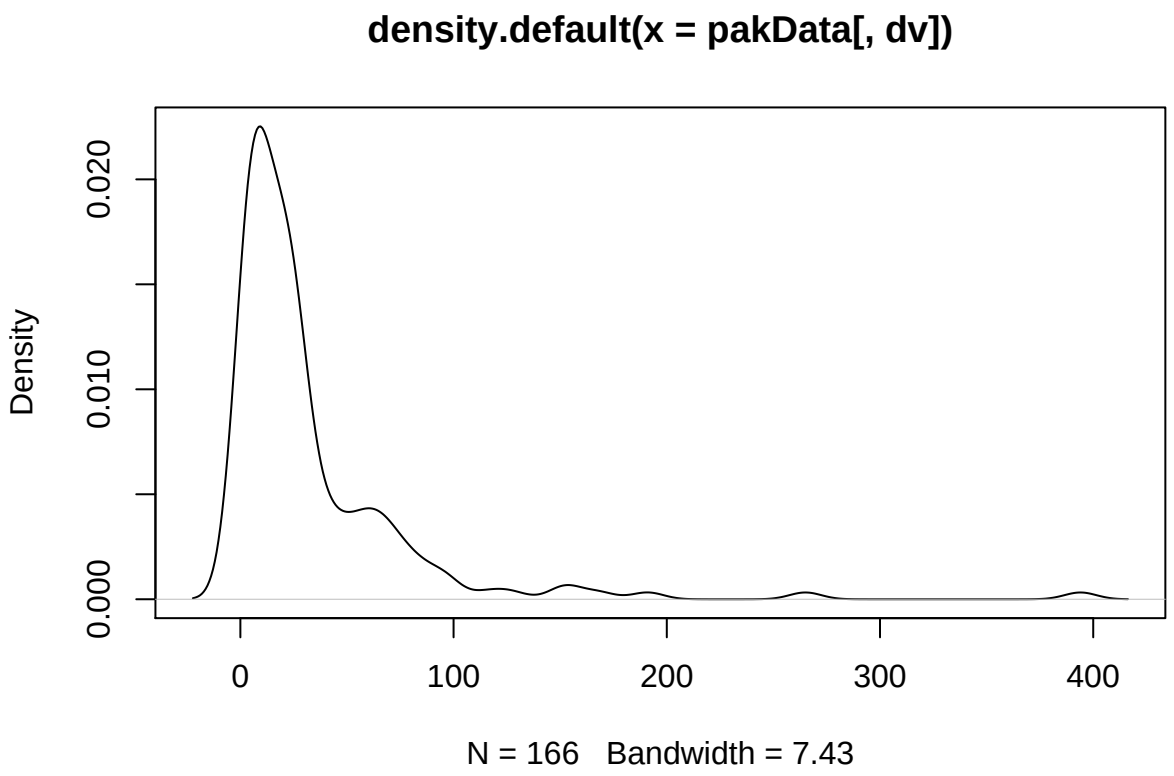
- As the number of observations trends toward infinity (or in practice, gets really big) then the Poisson is a good approximation of the Binomial.
- Similarly, if there is a fixed upper limit for your DV, use a Binomial instead.
- For particularly small n, the logistic regression model and the Poisson model will give similar results
- If counts span a range of values (e.g. they're in 'bins'), multinomial logistic regression should be used instead.

Let's work with some count data and run a model.

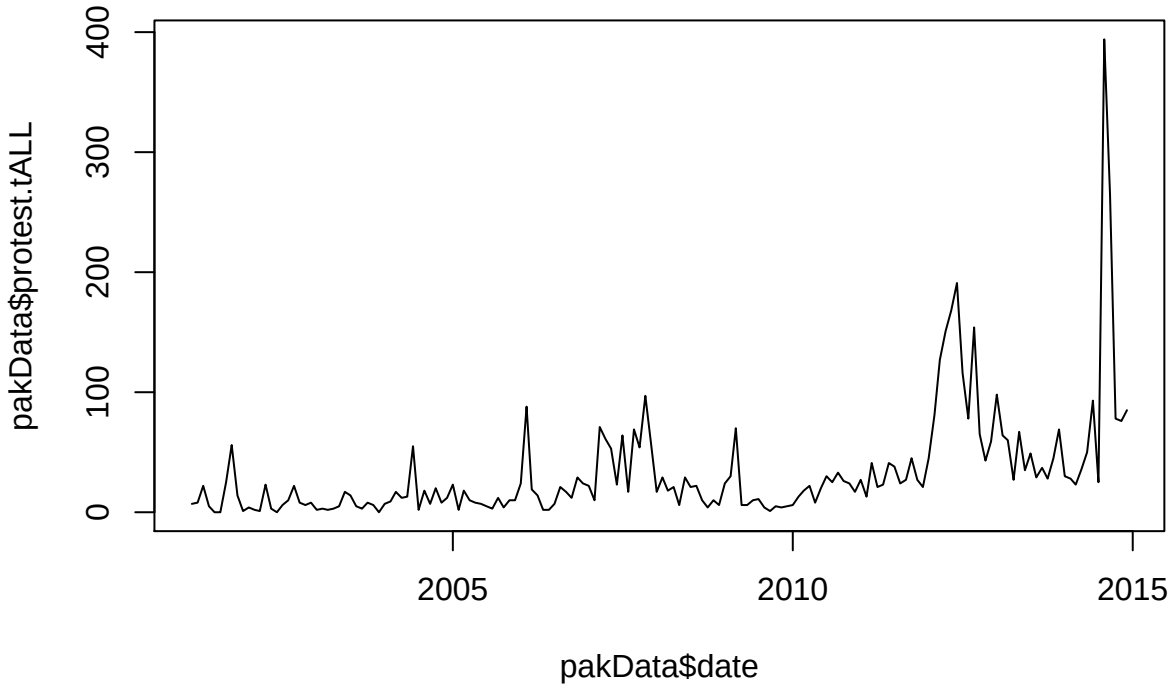
```
#####################################################
# Load Pakistan Protest Data
load(paste0(labPath, 'pakProtestData.rda')) # Loads object called pakData
#####################################################
```

Our data are about protests in Pakistan. We'll be modeling the number of protests based on some political and economic factors.

```
#####################################################
# Examine DV
dv <- 'protest.tALL'
plot( density( pakData[,dv] ) )
```




```
# Examine protests over time
plot(pakData$date, pakData$protest.tALL, type='l')
```




```
#######################################################

#######################################################
# Model count variables using a poisson process
ivs <- c(
    'ProxElection', # Proximity to election
    'NY.GDP.PCAP.KD.l1', # GDP per capita in constant 2000 dollars, lagged by 1 month
    'FP.CPI.TOTL.ZG.l1', # Inflation lagged by one month
    'intratension.l1', # Number of conflictual actions and events internal to the government, lagged by
    'W.centdist.std.protest.tALL.l1' # Spatial temporal lag of dv
    )
form <- as.formula( paste0(
    dv, ' ~ ',
    paste(ivs, collapse=' + ')
    ) )

summary(poisMod <- glm(form, data=pakData, family=poisson))
```

```
##
## Call:
## glm(formula = form, family = poisson, data = pakData)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -10.3143   -3.2954   -0.9565    1.2091   17.0074
##
```

```
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -8.972e+00  2.675e-01 -33.539  < 2e-16 ***
## ProxElection                     2.313e-05  5.829e-05   0.397    0.692
## NY.GDP.PCAP.KD.l1                1.706e-02  3.449e-04  49.468  < 2e-16 ***
## FP.CPI.TOTL.ZG.l1               -1.007e-01  4.237e-03 -23.777  < 2e-16 ***
## intratension.l1                 -2.227e-03  4.840e-04  -4.600 4.21e-06 ***
## W.centdist.std.protest.tALL.l1 -1.732e-01  1.093e-02 -15.849  < 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: 6786.0  on 165  degrees of freedom
## Residual deviance: 3166.8  on 160  degrees of freedom
## AIC: 3945.8
##
## Number of Fisher Scoring iterations: 5
```

Woo, stars everywhere! Interesting that proximity to an election doesn't matter, but all the other variables are significant and we'll look at some of these relationships in more detail. But first, is this even the right type of model to be running? Sure, we've got count data, but what about dispersion?

As we know, the Poisson distribution is interesting, in that its mean and variance are equal. In practice, this is rare. Particularly if there are large positive outliers in the DV, the variance will be greater than the mean. For such cases, we can use a Negative Binomial model instead.

Let's check the dispersion parameter, using the `dispersiontest()` function from the `AER` package. This is a hypothesis test for whether the mean and variance of the DV (found via a model object) are equal.

Overdispersion is a symptom of a variety of modeling challenges, but the most substantively important is that events that are positively correlated (previous events increase the rate of subsequent events) will manifest overdispersion in Poisson models. You'll also get incorrect standard error estimates.

```
# Check for overdispersion via Cameron and Trivedi
dispersiontest(poisMod)
```

```
##
##  Overdispersion test
##
## data:  poisMod
## z = 4.7232, p-value = 1.161e-06
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##   23.21046
```

We can also test for overdispersion by looking at the relationship of the standard deviation and the square root of the mean, which should also be equal (since their squares are equal)

```
# Gelman & Hill test for overdispersion
# standard deviation of the Poisson predictions is equal to the square root of the mean
predVals = data.matrix(cbind(1, pakData[,ivs])) %*% coef(poisMod)
predCnts = exp(predVals)
z=(pakData[,dv] - predCnts)/sqrt(predCnts)
df=nrow(pakData) - (length(ivs) + 1)
dispRat = sum(z^2)/df
print(paste0('Overdispersion ratio is ', round(dispRat,2) ) )
```

```
## [1] "Overdispersion ratio is 24.13"
```

```
pval = pchisq(sum(z^2), df, lower.tail = F)
print(paste0('p-value ', round(pval, 4)) )
```

```
## [1] "p-value 0"
```

```
# p-value is 0, indicating that the probability is essentially zero
# that a random variable from this chi sq distribution would be as large
# as what we found
#######################################################
```

There are a number of ways to deal with overdispersion. The first, which I mentioned above, is just to use a Negative Binomial model. You can also transform the standard errors of a Poisson model by multiplying them by the square root of the dispersion parameter (via the Gelman and Hill calculation above)

```
dispAdj <- sqrt(dispRat)

# Rebuild coefficient table
coefTable <- summary( poisMod )$'coefficient'
coefTable[,2] <- coefTable[,2] * dispAdj # Adjust standard errors
coefTable[,3] <- coefTable[,1]/coefTable[,2] # Recalculate z-statistic
coefTable[,4] <- 2*pnorm( -abs(coefTable[,3]) ) # Recalculate p values
```

Now we can compare the original model results with those using the dispersion parameter adjustment, and those from both a Negative Binomial and quasipoisson models (which are both used in cases of overdispersion).

```
# Original Model
print( round(summary( poisMod )$'coefficient', 3) )
```

```
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -8.972      0.268 -33.539    0.000
## ProxElection                      0.000      0.000   0.397    0.692
## NY.GDP.PCAP.KD.l1                 0.017      0.000  49.468    0.000
## FP.CPI.TOTL.ZG.l1                -0.101      0.004 -23.777    0.000
## intratension.l1                  -0.002      0.000  -4.600    0.000
## W.centdist.std.protest.tALL.l1   -0.173      0.011 -15.849    0.000
```

```
# Original Model with dispersion parameter adjustment
print(round(coefTable, 3))
```

```
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -8.972      1.314  -6.827    0.000
## ProxElection                      0.000      0.000   0.081    0.936
## NY.GDP.PCAP.KD.l1                 0.017      0.002  10.070    0.000
## FP.CPI.TOTL.ZG.l1                -0.101      0.021  -4.840    0.000
## intratension.l1                  -0.002      0.002  -0.936    0.349
## W.centdist.std.protest.tALL.l1   -0.173      0.054  -3.226    0.001
```

```
# Negative binomial model
negbinMod <- glm.nb(form, data=pakData)
round( summary(negbinMod)$'coefficient', 3 )
```

```
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -7.050      1.014  -6.952    0.000
## ProxElection                      0.000      0.000  -0.853    0.394
## NY.GDP.PCAP.KD.l1                 0.014      0.002   9.518    0.000
## FP.CPI.TOTL.ZG.l1                -0.100      0.019  -5.334    0.000
## intratension.l1                  -0.002      0.002  -0.661    0.508
```

```
## W.centdist.std.protest.tALL.l1   -0.087      0.056  -1.557    0.120
```

```
# Quasilikelihood
qpoisMod <- glm(form, data=pakData, family=quasipoisson)
round( summary(qpoisMod)$'coefficient', 3 )
```

```
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -8.972      1.314  -6.827    0.000
## ProxElection                      0.000      0.000   0.081    0.936
## NY.GDP.PCAP.KD.l1                 0.017      0.002  10.070    0.000
## FP.CPI.TOTL.ZG.l1                -0.101      0.021  -4.840    0.000
## intratension.l1                  -0.002      0.002  -0.936    0.350
## W.centdist.std.protest.tALL.l1   -0.173      0.054  -3.226    0.002
```

```
#######################################################
```

NOTE: The Quasi-Poisson uses a Poisson distribution, but also estimates a separate scale parameter (e.g. variance), even though in the regular Poisson this is fixed. I'm not sure whether Daniel will talk about these in depth, but I've just included this as another thing to compare against.

Substantively, it seems like conflictual government events and actions are not something we should look at for further analysis, even though the original model suggested we should. We may also want to be skeptical of whether the spatial-temporal lag of protests has a meaningful impact on protests, since this had a smaller effect in the NB model than others.

Another limitation of the standard coutn model is that the zeros and the nonzeros are assumed to come from the same data-generating process. Instead, it might be true that there are two separate processes here:

1. The decision to protest
2. How much to protest

We can remove this assumption (and hence potentially get a better picture of things) by using hurdle and zero-inflated models. These models are particularly useful with data where there are a LOT of zeros. Our data only have 4 zeros, so we're going to cheat a little bit and add some zeros.

```
# Introducing some more zeros
sum(pakData[,dv]==0)
```

```
## [1] 4
```

```
pakData2 = pakData
pakData2[ which(pakData2[,dv] < 4), dv] = 0
sum(pakData2[,dv]==0) #Meh, more-ish. More than 10% now
```

```
## [1] 19
```

```
# Hurdle & Zero-inflated models
Form2 = as.formula( paste0(
    dv, ' ~ ',
    paste(ivs[1:3], collapse=' + '), ' | ',
    paste(ivs[4:5], collapse=' + ')
    ) )
```

The distinction between hurdle and zero-inflated models. ### Hurdle models are for modeling 2 types of units: (1) units that never experience outcome and (2) units that always experience the outcome at least once ### Zero-inflated models are used for 2 types of units: (1) units that never experience outcome and (2) units that experience outcome but not always

They are similar in how they model the first stage: Both are modeling the probability of no event and some event. But they differ in the second stage: * Hurdle models use a zero-truncated probability distribution

function (e.g., zero truncated poisson, thus counts must be non-zero) * Zero inflated models use a typical discrete probability distribution (e.g., poisson, thus counts can still be zero)

```
hpoisMod <- hurdle(Form2, data=pakData, dist='poisson')
summary(hpoisMod)$'coefficient'
```

```
## $count
##                         Estimate   Std. Error     z value     Pr(>|z|)
## (Intercept)       -6.581351e+00 6.411589e-01 -10.2647739 1.015586e-24
## ProxElection       6.947987e-05 7.692269e-05   0.9032428 3.663970e-01
## NY.GDP.PCAP.KD.l1  1.329583e-02 7.192574e-04  18.4854922 2.702220e-76
## FP.CPI.TOTL.ZG.l1 -9.664896e-02 5.059249e-03 -19.1034200 2.364779e-81
##
## $zero
##                                   Estimate Std. Error    z value
## (Intercept)                     1.27523579 1.31136086  0.9724522
## intratension.l1                 0.09152781 0.04925178  1.8583655
## W.centdist.std.protest.tALL.l1 -0.14146682 0.64477745 -0.2194041
##                                   Pr(>|z|)
## (Intercept)                     0.33082562
## intratension.l1                 0.06311713
## W.centdist.std.protest.tALL.l1 0.82633527
```

```
zpoisMod <- zeroinfl(Form2, data=pakData, dist='poisson')
summary(zpoisMod)$'coefficient'
```

```
## $count
##                         Estimate   Std. Error     z value     Pr(>|z|)
## (Intercept)       -6.583214e+00 6.526422e-01 -10.0870189 6.305538e-24
## ProxElection       6.966886e-05 7.768629e-05   0.8967974 3.698271e-01
## NY.GDP.PCAP.KD.l1  1.329786e-02 7.320262e-04  18.1658234 9.625214e-74
## FP.CPI.TOTL.ZG.l1 -9.664283e-02 5.082374e-03 -19.0152921 1.274251e-80
##
## $zero
##                                   Estimate Std. Error    z value
## (Intercept)                     -1.28348252 1.31646153 -0.9749487
## intratension.l1                 -0.09222922 0.05008369 -1.8415021
## W.centdist.std.protest.tALL.l1   0.14806330 0.64953086  0.2279542
##                                   Pr(>|z|)
## (Intercept)                     0.32958568
## intratension.l1                 0.06554801
## W.centdist.std.protest.tALL.l1 0.81968182
```

```
######################################################
```

Split into groups of 2. Take a few minutes and:

1. Substantively interpret the results of one of the two-stage models.
2. Think about prediction with one-stage count models. Does it differ from what we've been doing in the past?