Generalized Linear Models, part 2

Frank Edwards

3/27/2020

Overdispersion

The Poisson model

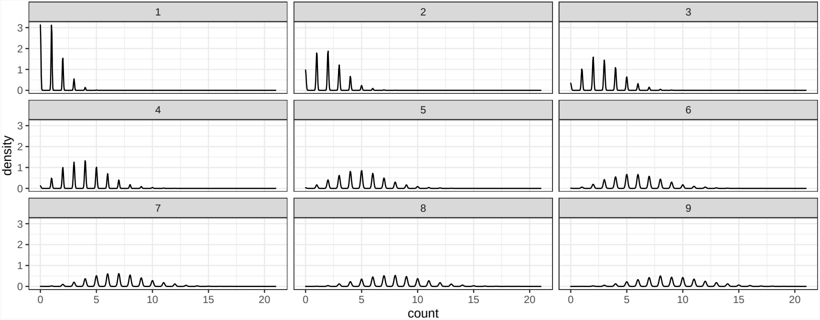
$$y \sim Poisson(\lambda)$$

$$E(y) = \lambda$$

$$Var(y) = \lambda$$

The Poisson distribution

```r
ggplot(pois_demo, aes(x=count)) +
  geom_density(adjust = 1/4) +
  facet_wrap(~lambda)
```



Overdispersion

A variable is said to be *overdispersed* if the variance exceeds what is expected by the model.

A count is overdispersed if it's variance exceeds it's mean (Poisson) or it's variance exceeds $np(1-p)$ (Binomial).

What overdispersion looks like in practice

```r
### on your computer, run fe<-read_csv("./slides/fe_demo.csv")
### make total deaths and pop per county
fe<-read_csv("./fe_demo.csv") %>%
  group_by(fips, state) %>%
  summarise(pop = sum(pop),
            deaths = sum(deaths))
head(fe)
```

```
## # A tibble: 6 x 4
## # Groups:   fips [6]
##     fips state   pop deaths
##    <dbl> <chr> <dbl>  <dbl>
## 1  1001 AL    19326      0
## 2  1003 AL    71551      6
## 3  1005 AL    11292      2
## 4  1007 AL     9297      1
## 5  1009 AL    21246      2
```

Estimating an intercept-only offset Poisson model

$$d_i \sim Poisson(\lambda)$$

$$\log \lambda_i = \log p_i + \alpha$$

$$\alpha \sim Normal(0, 2)$$

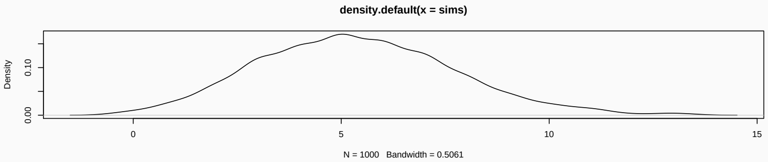
```
fe_pois_offset<-ulam(alist(
  deaths ~ dpois(l),
  log(l) <- log(pop) + a,
  a ~ dnorm(0,2)
), data = fe, chains = 4, cores = 2)
```

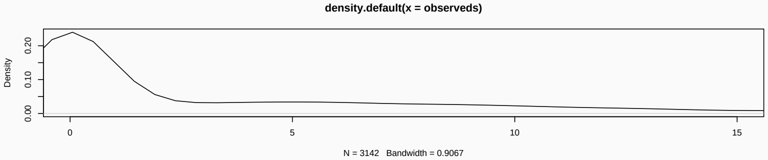
Let's compare model predictions to the observed data

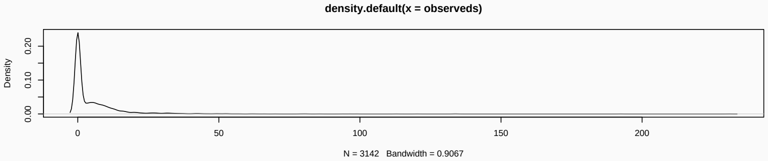
```r
### compare the model to the observed
sim_dat<-data.frame(pop = 1e5)
sims<-sim(fe_pois_offset, data = sim_dat)
observeds<-fe$deaths / fe$pop * 1e5
tab_dat<-data.frame(type = c("observed", "model"),
                    var_y = c(var(observeds), var(sims)),
                    mean_y = c(mean(observeds), mean(sims)))
library(pander)
pander(tab_dat)
```

type	var_y	mean_y
observed	131.2	5.08
model	5.44	5.362

Let's compare model predictions to the observed data







Overdispersion in practice

- Overdispersion results from mixed processes, when outcomes have more than one cause or are a sum or product of processes
- Most count variables are overdispersed
- Failing to adjust for overdispersion leads to over-confidence in posterior inference (intervals are too narrow)
- My advice: assume overdispersion

Addressing overdispersion

- Mixture models gamma + Poisson (negative binomial) and beta + binomial (beta binomial) models handle this problem well
- Multilevel models with observation-level intercepts also effectively model overdispersion

Modeling overdispersion: the negative binomial model

- To allow for unmeasured variables to influence counts, we can let each Poisson variable have it's own event rate
- We mix the Poisson and the gamma distributions to model the count (Poisson) and the rate (gamma) of each observation
- This adds one extra parameter to the model: a dispersion parameter ϕ that allows for extra variance

$$y \sim \text{Negative Binomial}(\lambda, \phi)$$

Re-estimating the police deaths model

$$d_i \text{Negative Binomial}(\lambda, \phi)$$

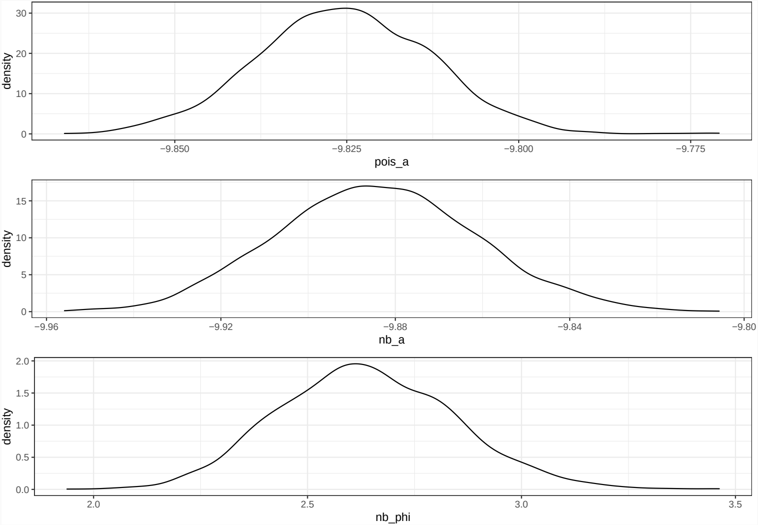
$$\log \lambda_i = \log p_i + \alpha$$

$$\alpha \sim Normal(0, 2)$$

$$\phi \sim Exponential(1)$$

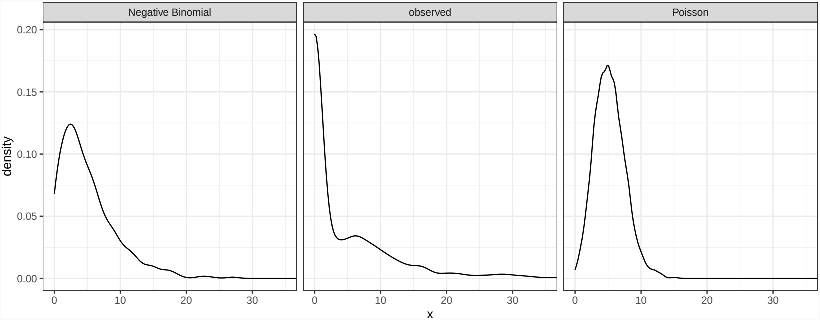
```
fe_NB_offset<-ulam(alist(
  deaths ~ dgampois(l, p),
  log(l) <- log(pop) + a,
  a ~ dnorm(0,2),
  p ~ dexp(1)
), data = fe, chains = 4, cores = 2)
```

Examining the posteriors



How Negative Binomial influences inference

```
sims_pois<-sim(fe_pois_offset, sim_dat)
sims_nb<-sim(fe_NB_offset, sim_dat)
plot_dat<-data.frame(
  param =
    rep(c("Poisson", "Negative Binomial", "observed"),
        each = 1000),
  x = c(sims_pois, sims_nb, sample(observeds, 1000))
)
ggplot(plot_dat, aes(x = x)) +
  geom_density() +
  facet_wrap(~param) +
  coord_cartesian(xlim=c(0, 35))
```



Zero inflation

When there are more zeroes than your model expects

- Sometimes multiple processes are at play in count models: 1) determines whether the event occurs at all, and if so 2) determines the count of events
- If we are counting salamander populations in a forest, ordinary ecological variables like forest cover may predict population sizes. But another process, like pollutant exposure may determine whether there are any salamanders there at all.

The zero inflated model

This model contains two components.

1. A model describing the probability that the outcome is zero
2. A model describing the expected event count

Remember, regular count models also produce zeroes!

Possible zero inflated count outcomes

- Number of snow days in a school district
- Prison admissions in a county
- Well-visits to a doctor per person

How many fish did you catch on the camping trip?

I'm guessing that people who went camping along were more likely to go fisihng. People in big groups and groups with kids were less likely to go fishing at all.

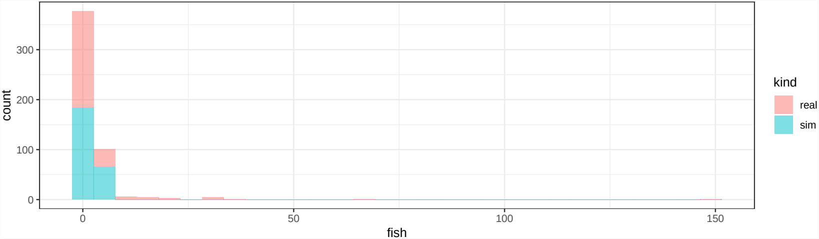
```
zinb <- read.csv("https://stats.idre.ucla.edu/stat/data/fish.csv")
  select(persons, camper, child, count)
head(zinb)
```

```
##   persons camper child count
## 1       1      0     0     0
## 2       1      1     0     0
## 3       1      0     0     0
## 4       2      1     1     0
## 5       1      0     0     1
## 6       4      1     2     0
```

Let's think about this

Let's assume that the average person who goes fishing catches 2 fish. That would give us a Poisson(2) distribution.

```
sim_fish<-rpois(250, 2)
plot_dat<-data.frame(fish = c(sim_fish, zinb$count),
                     kind = rep(c("sim", "real"), each = 250))
ggplot(plot_dat,
       aes(x = fish, fill = kind)) +
  geom_histogram(alpha = 0.5)
```



Let's build our zero-inflated model

We suggest that kids and other people get in the way of fishing. If you go fishing, more people (rods) likely means more fish.

- First, we will run a logistic regression to estimate the probability that each group went fishing at all.
- We add this probability of not going fishing to our Poisson estimated probability of catching nothing when people did go fishing.
- Because these are separate linear models, we'll get different parameters

$$fish_i \sim \text{Zero-inflated Poisson}(p_i, \lambda_i)$$
$$\text{logit}(p_i) = \alpha_p + \beta_{1p} \times kids_i + \beta_{2p} \times persons_i$$
$$\log(\lambda_i) = \alpha_l + \beta_l \times persons_i$$

The model

$$fish_i \sim \text{Zero-inflated Poisson}(p_i, \lambda_i)$$

$$\text{logit}(p_i) = \alpha_p + \beta_{1p} \times kids_i + \beta_{2p} \times persons_i$$

$$\log(\lambda_i) = \alpha_l + \beta_l \times persons_i$$

$$\alpha_p \sim Normal(0, 2)$$

$$\beta_{1p} \sim Normal(0, 2)$$

$$\beta_{2p} \sim Normal(0, 2)$$

$$\beta_{1l} \sim Normal(0, 2)$$

$$\alpha_l \sim Normal(0, 2)$$

Estimating the model

```
mfish<-ulam(alist(
  count ~ dzipois(p, lambda),
  logit(p) <- ap + b1p * child + b2p * persons,
  log(lambda)<- al + b1l * persons,
  ap ~ dnorm(0, 2),
  b1p ~ dnorm(0, 2),
  b2p ~ dnorm(0, 2),
  al ~ dnorm(0, 2),
  b1l ~ dnorm(0, 2)
), data = zinb, chains = 4, cores = 4)
```

Let’s look at this ridiculous creature we’ve created

```r
precis(mfish)
```

```
##           mean         sd       5.5%      94.5%     n_eff    Rhat4
## ap   1.0725703 0.39889310  0.4499551  1.7138247 1112.4819 1.000467
## b1p  2.0693453 0.30030476  1.6148011  2.5575466 1068.7015 1.001920
## b2p -0.8842018 0.17967596 -1.1770035 -0.6062897 1039.5755 1.001042
## al  -0.2990006 0.15416836 -0.5454430 -0.0439894  871.4682 1.008176
## b1l  0.7514892 0.04304351  0.6813428  0.8190742  865.7324 1.008047
```

Ordered categorical outcomes

Categorical variables and ordered categorical variables

- Categorical variables have no inherent rank (i.e. States)
- Ordered categorical variables have a clear sequence or ordering

Ordered categorical variables

- Likert scales (1. Strongly disagree … 7. Strongly agree)
- Passenger class
- School grade
- Educational attainment (less than HS, HS diploma, Some college, college degree, graduate degree)

Ordered categorical variables

- Likert scales (1. Strongly disagree … 7. Strongly agree)
- Passenger class
- School grade
- Educational attainment (less than HS, HS diploma, Some college, college degree, graduate degree)

While these are often modeled as continuous, that assumes symmetric distances between ranks.

How likely are you to apply to grad school?

```r
library(haven)
dat <- read_dta("https://stats.idre.ucla.edu/stat/data/ologit.dta")
head(dat)
```

```
## # A tibble: 6 x 4
##                 apply pared public   gpa
##             <dbl+lbl> <dbl>  <dbl> <dbl>
## 1 2 [very likely]         0      0  3.26
## 2 1 [somewhat likely]     1      0  3.21
## 3 0 [unlikely]            1      1  3.94
## 4 1 [somewhat likely]     0      0  2.81
## 5 1 [somewhat likely]     0      0  2.53
## 6 0 [unlikely]            0      1  2.59
```

Cumulative proportions

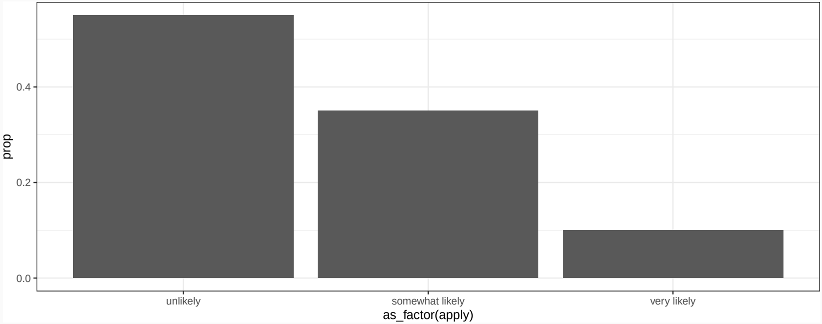
```
plot_dat<-dat %>%
  group_by(apply) %>%
  summarise(n = n()) %>%
  ungroup() %>%
  mutate(prop = n/sum(n),
         cum_prop = cumsum(prop))

plot_dat
```

```
## # A tibble: 3 x 4
##                  apply     n  prop cum_prop
##              <dbl+lbl> <int> <dbl>    <dbl>
## 1 0 [unlikely]          220  0.55     0.55
## 2 1 [somewhat likely]   140  0.35     0.9
## 3 2 [very likely]        40  0.1      1
```

Visualizing proportions

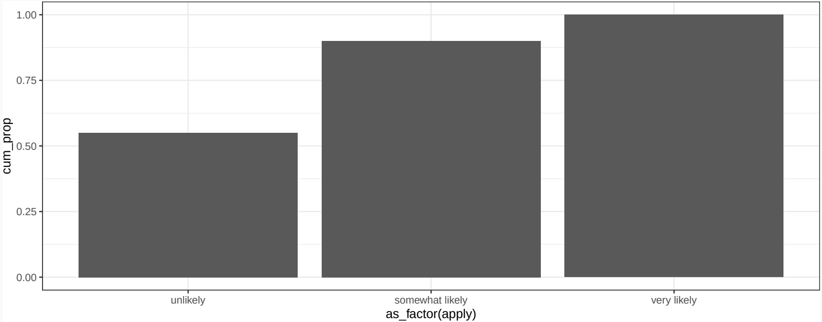
```r
ggplot(plot_dat,
       aes(x = as_factor(apply), y = prop)) +
  geom_col()
```



Visualizing cumulative proportions

Note that if we have 3 categories, we only need values for 2 to know all 3, because of the law of total probability.

```
ggplot(plot_dat,
       aes(x = as_factor(apply), y = cum_prop)) +
  geom_col()
```



The ordered-logit model

We can allow the distance between each group to be uneven with a cumulative link function. The probability of being in group k is relative to the prior group. We then use a logit linear model for each level of the outcome.

$$apply_i \sim Categorical(p)$$

$$p_1 = q_1$$

$$p_2 = q_2 - q_1$$

$$p_3 = 1 - q_2$$

$$\text{logit}(q_k) = \kappa_k - \phi_i$$

$$\phi_i = \text{linear model}$$

And your priors

Let's build a model!

I think that people with higher GPAs are more likely to say they will apply to grad school (controversial).

We can write our model more compactly as

$$apply_i \sim \text{Ordered-logit}(\phi_i, \kappa)$$

$$\phi_i = \beta \times \text{GPA}_i$$

$$\kappa_k \sim Normal(0, 2)$$

$$\beta \sim Normal(0, 2)$$

Estimate the model

```r
d_slim<-dat %>%
  mutate(apply = as.numeric(apply) + 1) %>%
  select(apply, gpa)

m_ord<-ulam(alist(
  apply ~ dordlogit(phi, kappa),
  phi <- b * gpa,
  b ~ dnorm(0,2),
  kappa ~ dnorm(0,2)
), data = d_slim, cores = 4, chains = 4)
```

Checking the posterior

```r
precis(m_ord, depth = 2)
```

```
##                mean        sd      5.5%     94.5%    n_eff    Rhat4
## b        0.4765858 0.2178227 0.1424147 0.8271994 419.1864 1.010764
## kappa[1] 1.6210121 0.6620412 0.6031219 2.6702127 413.1166 1.011807
## kappa[2] 3.6301744 0.6802613 2.5926215 4.7128605 427.0600 1.010246
```

Posterior inference

```
sim_dat<-data.frame(gpa = seq(2, 4, by = 1))
sims<-sim(m_ord, sim_dat)
sims<-as.data.frame(sims)
names(sims)<-c("GPA2", "GPA3", "GPA4")
table(sims$GPA2)
```

```
##
##   1   2   3
## 668 267  65
```

```
table(sims$GPA3)
```

```
##
##   1   2   3
## 565 342  93
```

```
table(sims$GPA4)
```

```
##
##   1   2   3
## 429 441 130
```

Summary

- Mixture models are very useful for real-world processes
- Counts are often (always) over-dispersed
- Counts often have multiple procesess at play and excess zeroes
- Categoricals are often ordered, and have unequally spaced differences
- HW: 12E1 - 12E4; 12H6 if you want practice