CHAPTER

14

Overdispersion, Zero-Inflation, and Offsets in the GLM

OUTLINE

Two features specific to nonnormal generalized linear models (GLMs) are overdispersion and offsets. Zero-inflation can be called a specific form of overdispersion: there are more zeroes than expected. Here, we briefly deal with each of them in the context of the hare counts example.

14.1 OVERDISPERSION

14.1.1 Introduction

In both distributions commonly used to model counts (Poisson and binomial), the dispersion (the variability in the counts) is not a free parameter but instead is a function of the mean. The variance is equal to the mean (λ) for the Poisson and equal to the mean ($N * p$) times $1 - p$ for the binomial distribution (see Chapters 17–19). This means that *for a Poisson or binomial random variable, the models for the counts come with a "built-in" variability* and the magnitude of that variability is known. In an analysis of deviance conducted in a classical statistical analysis of the model, the residual deviance of the model will be about the same magnitude as the residual degrees of freedom, i.e., the mean deviance ratio (= residual deviance/residual df) is about 1.

However, in real life, count data are almost always more variable than expected under the Poisson or binomial models. This is called overdispersion or extra-Poisson or extra-binomial variation and means that the residual variation is larger than prescribed by a Poisson or binomial. Overdispersion can occur because there are hidden correlations that have not been included in the model, e.g., when individuals in family groups are assumed to be independent, or when important covariates have not been included. When overdispersion is not modeled, tests and confidence intervals will be overconfident (although means won't normally be biased). Therefore, overdispersion should be tested and corrected for when necessary.

The simplest way to correct for overdispersion in a classical analysis is by the quasi-likelihood (McCullagh and Nelder, 1989) and by using `family=quasipoisson` (or `quasibinomial`) in the R function `glm()`. Using WinBUGS, there are several ways in which one can account for overdispersion. One is to specify a distribution that is overdispersed relative to the Poisson, such as the negative binomial. Another solution, and the one we illustrate here, is to add into the linear predictor for the Poisson intensity a normally distributed random effect. Technically, this model is then a Poisson generalized linear mixed model (GLMM; see Chapter 16 for a more formal introduction). It is sometimes called a Poisson-lognormal model (Millar 2009).

14.1.2 Data Generation

We generate a slightly modified hare count data set, where in addition to the land-use difference in mean density, there is also a normally distributed site-specific effect in the linear predictor. For illustrative purposes, we also generate a sister data set without overdispersion.

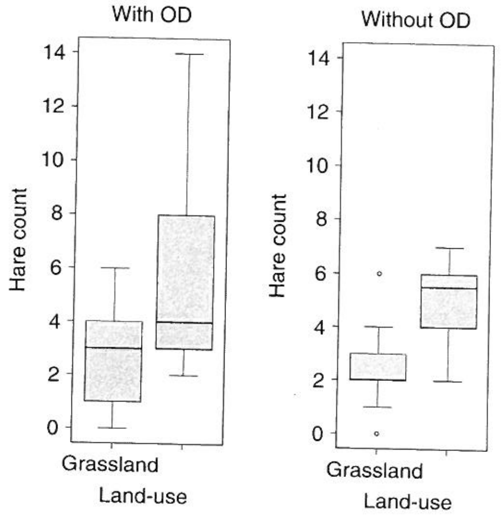

FIGURE 14.1 Hare counts by land-use with and without overdispersion (OD) Overdispersion was caused by site-specific differences in hare density.

```
n.site <- 10
x <- gl(n = 2, k = n.site, labels = c("grassland", "arable"))
eps <- rnorm(2*n.site, mean = 0, sd = 0.5)                    # Normal random effect
lambda.OD <- exp(0.69 +(0.92*(as.numeric(x)-1) + eps) )
lambda.Poisson <- exp(0.69 +(0.92*(as.numeric(x)-1)) ) # For comparison
```

We add the noise that comes from a Poisson and inspect the hare counts we've generated (Fig. 14.1):

```
C.OD <- rpois(n = 2*n.site, lambda = lambda.OD)
C.Poisson <- rpois(n = 2*n.site, lambda = lambda.Poisson)

par(mfrow = c(1,2))
boxplot(C.OD ~ x, col = "grey", xlab = "Land-use", main = "With OD", ylab = "Hare
count", las = 1, ylim = c(0, max(C.OD)))
boxplot(C.Poisson ~ x, col = "grey", xlab = "Land-use", main = "Without OD", ylab =
"Hare count", las = 1, ylim = c(0, max(C.OD)) )
```

14.1.3 Analysis Using R

We conduct a classical analysis of the overdispersed data once without and then with correction for overdispersion (using `glm(, family = quasi...)`).

```
glm.fit.no.OD <- glm(C.OD ~ x, family = poisson)
glm.fit.with.OD <- glm(C.OD ~ x, family = quasipoisson)
summary(glm.fit.no.OD)
summary(glm.fit.with.OD)
anova(glm.fit.no.OD, test = "Chisq")
anova(glm.fit.with.OD, test = "F")

> summary(glm.fit.no.OD)

Call:
glm(formula = C.OD ~ x, family = poisson)

[ ... ]

Coefficients:
              Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)    0.9933      0.1925    5.161   2.46e-07 ***
xarable        0.7646      0.2330    3.282   0.00103  **
- - -
[ ... ]

(Dispersion parameter for poisson family taken to be 1)

[ ... ]

> summary(glm.fit.with.OD)

Call:
glm(formula = C.OD ~ x, family = quasipoisson)

[ ... ]

Coefficients:
              Estimate   Std. Error    t value     Pr(>|t|)
(Intercept)    0.9933       0.2596      3.826      0.00124   **
xarable        0.7646       0.3143      2.433      0.02563   *
- - -
[ ... ]

(Dispersion parameter for quasipoisson family taken to be 1.819569)

    Null deviance:  44.198  on 19  degrees of freedom
Residual deviance:  32.627  on 18  degrees of freedom
[ ... ]

> anova(glm.fit.no.OD, test = "Chisq")
Analysis of Deviance Table

Model: poisson, link: log

[ ... ]

      Df  Deviance    Resid. Df    Resid. Dev   P(>|Chi|)
NULL                         19        44.198
x      1    11.571           18        32.627        0.001
> anova(glm.fit.with.OD, test = "F")
Analysis of Deviance Table
```

```
Model: quasipoisson, link: log

[ ... ]

      Df  Deviance   Resid. Df   Resid. Dev        F     Pr(>F)
NULL                        19       44.198
x      1    11.571          18       32.627   6.3591   0.02132 *
```

Thus, the parameter estimates don't change when accounting for overdispersion, but tests and standard errors do.

14.1.4 Analysis Using WinBUGS

In WinBUGS, it is easy to get from the simple Poisson t-test with homogeneous (Poisson) variance in the last chapter to the overdispersed Poisson t-test represented by the Poisson-lognormal model.

```
# Define model
sink("Poisson.OD.t.test.txt")
cat("
model {

# Priors
 alpha ~ dnorm(0,0.001)
 beta ~ dnorm(0,0.001)
 sigma ~ dunif(0, 10)
 tau <- 1 / (sigma * sigma)

# Likelihood
 for (i in 1:n) {
    C.OD[i] ~ dpois(lambda[i])
    log(lambda[i]) <- alpha + beta *x[i] + eps[i]
    eps[i] ~ dnorm(0, tau)
 }
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(C.OD = C.OD, x = as.numeric(x)-1, n = length(x))

# Inits function
inits <- function(){ list(alpha=rlnorm(1), beta=rlnorm(1), sigma = rlnorm(1))}

# Parameters to estimate
params <- c("lambda","alpha", "beta", "sigma")
```

Note that as soon as we start estimating variances (here, of the overdispersion effects eps), we need longer chains.

```
# MCMC settings
nc <- 3              # Number of chains
ni <- 3000           # Number of draws from posterior per chain
nb <- 1000           # Number of draws to discard as burn-in
nt <- 5              # Thinning rate

# Start Gibbs sampling
out <- bugs(data=win.data, inits=inits, parameters.to.save=params,
model.file="Poisson.OD.t.test.txt", n.thin=nt, n.chains=nc, n.burnin=nb, n.iter=ni,
debug = TRUE)

print(out, dig = 3)
```

14.2 ZERO-INFLATION

14.2.1 Introduction

Zero-inflation can be called a specific form of overdispersion and is frequently found in count data. It means that there are more zeroes than expected under the assumed (e.g., Poisson or binomial) distribution. In the context of our hare counts, a typical explanation for excess zeroes is that some sites are simply not suitable for hares, such as paved parking lots, roof tops, or lakes; hence, resulting counts must be zeroes. In the remaining suitable sites, counts vary according to the assumed distribution. Thus, we may imagine a sequential genesis of zero-inflated counts: first, Nature determines whether a site may be occupied at all, and second, she selects the counts for those that are habitable in principle. Regression models that account for this kind of overdispersion are often called zero-inflated Poisson (ZIP) or zero-inflated binomial (ZIB) models.

A ZIP model for count C_i at site i can be written algebraically like this:

$$w_i \sim \text{Bernoulli}(\psi_i) \qquad \text{Suitability of a site} \qquad (14.1)$$

$$C_i \sim \text{Poisson}(w_i \times \lambda_i) \qquad \text{Observed counts} \qquad (14.2)$$

For each site i, Nature flips a coin that lands heads (i.e., $w_i = 1$) with probability ψ_i. We can't observe w_i perfectly, i.e., it is a latent or random effect. Only for sites with $w_i = 1$, Nature then rolls her Poisson (λ_i) die to determine the count C_i at that site. For sites with $w_i = 0$, the Poisson mean is $0 \times \lambda_i = 0$ and the corresponding Poisson die produces zero counts only.

We see that a ZIP model simply represents a set of two coupled GLMs: the logistic regression describes the suitability in principle of a site while the Poisson regression describes the variation of counts among suitable sites, i.e., those with $w_i = 1$. All the usual GLM features apply, and in particular, both the Bernoulli and the Poisson parameter can be expressed as a function of covariates on the link scale. These covariates may or may not be the same for both regressions.

I make four notes on the ZIP model in our context of hare counts: First, the model allows for two entirely different kinds of zero counts; those coming from the Bernoulli and those from the Poisson process. The former are zero counts at unsuitable sites while the latter are due to Poisson chance, i.e., for them, Nature's Poisson die happened to yield a zero. The actual distribution of an organism, i.e., the proportion of sites that is occupied (has nonzero counts), is a function of both processes. Hence, it would be wrong to say that Eqn. 14.1 describes the distribution and Eqn. 14.2 the abundance.

Second, the above ZIP model is a hierarchical, or random-effects, model with binary instead of normal random effects. It is an example of the kind of nonstandard GLMMs that are featured extensively in Chapters 20 and 21. There, we will see the site-occupancy species distribution model, another kind of zero-inflated GLM, but one where a Bernoulli or binomial distribution is zero-inflated with another Bernoulli, so we get a zero-inflated binomial (ZIB) model.

Third, some authors advocate ZIP models widely for inference about count data (Martin et al., 2005; Joseph et al., 2009). However, on ecological grounds, they appear most adequate in situations where *unknown* environmental covariates determine the suitability of a site. If covariates are known and have been measured, they are probably best added to the linear model for the Poisson mean. Distribution, or occurrence, is fundamentally a function of abundance, i.e., a species occurs at all sites where abundance is greater than zero. It appears contrived to model distribution completely separately from abundance.

Fourth, there is a variant of a ZIP model called the hurdle model (Zeileis et al., 2008), where the first step in the hierarchical genesis of the counts is assumed to be the same as in a ZIP model, i.e., $w_i \sim \text{Bernoulli}(\psi_i)$. But then, counts at suitable sites (i.e., with $w_i = 1$) are modeled as coming from a zero-truncated Poisson distribution, i.e., a Poisson for values excluding zero. Hurdles (thresholds) other than zero are also possible. Superficially, this model may appear "better" than a ZIP model because it only allows one kind of zero: that coming from the Bernoulli process. However, it posits that all sites that are suitable in principle *will* be occupied and have a count greater than 0. This is not sensible biologically becausė, in reality, a suitable site may well be unoccupied as a result of local extinction, dispersal limitation, or some other reason.

14.2.2 Data Generation

We generate the simplest kind of zero-inflated count data for our (Poisson) hare example. We assume different densities in arable and grassland areas and a constant zero inflation, i.e., a single value of ψ for all sites, regardless of land-use or other environmental covariates.

```
psi <- 0.8
n.site <- 20
x <- gl(n = 2, k = n.site, labels = c("grassland", "arable"))
```

For each site, we flip a coin to determine its suitability and store the result in the latent state w_i.

```
w <- rbinom(n = 2*n.site, size = 1, prob = psi)
```

We assume identical effects of arable and grass as before and generate expected counts at suitable sites as before.

```
lambda <- exp(0.69 +(0.92*(as.numeric(x)-1)) )
```

We then add up (actually, multiply) the effects of both processes (Bernoulli and Poisson) and inspect the counts we've generated. Note how all counts at unsuitable sites (with $w_i = 0$) are zero.

```
C <- rpois(n = 2*n.site, lambda = w *lambda)
cbind(x, w, C)
```

14.2.3 Analysis Using R

A wide range of ZIP and related models can be fitted in R using the function `zeroinfl()` in package pscl; see, for instance, Zeileis et al. (2008). We load that package and fit the simplest possible ZIP model.

```
library(pscl)
fm <- zeroinfl(C ~ x | 1, dist = "poisson")

summary(fm)
> summary(fm)

Call:
zeroinfl(formula = C ~ x | 1, dist = "poisson")

Count model coefficients (poisson with log link):
            Estimate  Std. Error  z value   Pr(>|z|)
(Intercept)   0.7441      0.1773    4.197  2.71e-05 ***
xarable       0.8820      0.2095    4.209  2.56e-05 ***

Zero-inflation model coefficients (binomial with logit link):
            Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)  -1.6786      0.4943   -3.396  0.000684 ***
- - -
Signif. codes: 0'***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 8
Log-likelihood: -78.4 on 3 Df
```

Because of sampling and estimation error, the coefficients for the count model (corresponding to Eqn. 14.2) may not always be very close to the input values. Also note that what pscl calls the coefficient in the zero-inflation model corresponds to $1 - \psi$ in Eqn. 14.1. Typing `plogis (-1.6786)` in R convinces us that the function is doing what it should do.

14.2.4 Analysis Using WinBUGS

Next, the solution in WinBUGS. As always, the elementary manner of model specification using the BUGS language makes it very clear what model is fitted. To make the parameter estimates directly comparable, we also add a line that computes the logit of the zero-inflation parameter in R from the parameter ψ that we use here.

```
# Define model
sink("ZIP.txt")
cat("
model {

# Priors
 psi ~ dunif(0,1)
 alpha ~ dnorm(0,0.001)
 beta ~ dnorm(0,0.001)

# Likelihood
 for (i in 1:n) {
    w[i] ~ dbern(psi)
    C[i] ~ dpois(eff.lambda[i])
    eff.lambda[i] <- w[i]*lambda[i]
    log(lambda[i]) <- alpha + beta *x[i]
 }

# Derived quantity
 R.lpsi <- logit(1-psi)
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(C = C, x = as.numeric(x)-1, n = length(x))

# Inits function
inits <- function(){ list(alpha=rlnorm(1), beta=rlnorm(1), w = rep(1, 2*n.site))}
```

We will also estimate the latent state w_i, i.e., the intrinsic suitability for brown hares at each site.

```
# Parameters to estimate
params <- c("lambda","alpha", "beta", "w", "psi", "R.lpsi")

# MCMC settings (need fairly long chains)
nc <- 3                # Number of chains
ni <- 50000            # Number of draws from posterior per chain
nb <- 10000            # Number of draws to discard as burn-in
nt <- 4                # Thinning rate

# Start WinBUGS
out <- bugs(data=win.data, inits=inits, parameters.to.save=params,
model.file="ZIP.txt", n.thin=nt,n.chains=nc,n.burnin=nb, n.iter=ni, debug = TRUE)

print(out, dig = 3)
> print(out, dig = 3)
Inference for Bugs model at "ZIP.txt", fit using WinBUGS,
 3 chains, each with 50000 iterations (first 10000 discarded), n.thin = 4
 n.sims = 30000 iterations saved
         mean    sd   2.5%    25%    50%    75%  97.5%  Rhat  n.eff
[ ...]
alpha    0.733 0.175  0.384  0.617  0.736  0.853  1.066 1.001  11000
beta     0.884 0.208  0.479  0.744  0.883  1.024  1.300 1.001  30000
[ ...]
psi      0.827 0.064  0.689  0.786  0.831  0.873  0.936 1.001  30000
R.lpsi  -1.635 0.484 -2.684 -1.931 -1.596 -1.302 -0.795 1.001  30000
```

We find pretty similar estimates between R and WinBUGS.

14.3 OFFSETS

14.3.1 Introduction

In a Poisson GLM, we assume that the expected counts are adequately described by the effect of the covariates in the model. However, frequently, we have that the "counting window" is not constant, e.g., that study areas don't have the same size or, in temporal samples, that the duration of counting periods differ. To account for this known component of variation in the conditional Poisson mean, we define the log of the size of the "counting window" (study area size, count duration) as an *offset*. Effectively, we then model density as a response.

Let's consider this for the hare counts using algebra. The Poisson GLM is $C_i \sim \text{Poisson}(\lambda_i)$, i.e., hare counts C_i are conditionally distributed as Poisson with expected count λ_i. When study areas differ in size, we have $C_i \sim \text{Poisson}(A_i * \lambda_i)$, where A_i is the area of study area i. Therefore, the linear predictor becomes $\log(A_i * \lambda_i) = \log(A_i) + \log(\lambda_i)$. If we also wish to model a covariate x into the mean, we get $\log(A_i * \lambda_i) = \log(A_i) + \alpha + \beta * x_i$. This is equivalent to forcing the coefficient of log(area) to be equal to 1. That is, we effectively fit the model $\log(A_i * \lambda_i) = \beta_0 * \log(A_i) + \alpha + \beta * x_i$ with $\beta_0 = 1$. The offset compensates for the additional and known variation in the response resulting from differing study area size.

14.3.2 Data Generation

```
n.site <- 10
A <- runif(n = 2*n.site, 2,5)      # Area's range in size from 2 to 5 km2
x <- gl(n = 2, k = n.site, labels = c("grassland", "arable"))
linear.predictor <- log(A) + 0.69 +(0.92*(as.numeric(x)-1))
lambda <- exp(linear.predictor)
C <- rpois(n = 2*n.site, lambda = lambda)  # Add Poisson noise
```

14.3.3 Analysis Using R

We use R for an analysis with and without consideration of the differing areas.

```
glm.fit.no.offset <- glm(C ~ x, family = poisson)
glm.fit.with.offset <- glm(C ~ x, family = poisson, offset = log(A))
summary(glm.fit.no.offset)
summary(glm.fit.with.offset)
anova(glm.fit.with.offset, test = "Chisq")          # LRT
```

Comparing the residual deviance of the two models makes clear that specification of an offset represents a sort of correction for a systematic kind of overdispersion.

14.3.4 Analysis Using WinBUGS

Note how simple it is in WinBUGS to jump from one kind of analysis for the hare counts to another.

```
# Define model
sink("Offset.txt")
cat("
model {

# Priors
 alpha ~ dnorm(0,0.001)
 beta ~ dnorm(0,0.001)

# Likelihood
 for (i in 1:n) {
      C[i] ~ dpois(lambda[i])
      log(lambda[i]) <- 1 * logA[i] + alpha + beta *x[i]   # Note offset
   }
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(C = C, x = as.numeric(x)-1, logA = log(A), n = length(x))

# Inits function
inits <- function(){ list(alpha=rlnorm(1), beta=rlnorm(1))}

# Parameters to estimate
params <- c("lambda","alpha", "beta")

# MCMC settings
nc <- 3              # Number of chains
ni <- 1100           # Number of draws from posterior
nb <- 100            # Number of draws to discard as burn-in
nt <- 2              # Thinning rate

# Start Gibbs sampling
out <- bugs(data=win.data, inits=inits, parameters.to.save=params,
model.file="Offset.txt", n.thin=nt,n.chains=nc,n.burnin=nb, n.iter=ni, debug =
TRUE)

print(out, dig = 3)
```

14.4 SUMMARY

Overdispersion, zero-inflation, and offsets are important GLM topics. The specification of the associated models in WinBUGS is fairly easy and clarifies the actual meaning of these three topics. This is not usually the case when fitting these models in a canned routine in R or another software. Thus, this is another example of where the simple model specification in the BUGS language enforces an understanding of the fitted model that is easily lost in other stats packages.

EXERCISES

1. *Estimating a coefficient for an offset covariate*: Forcing the coefficient of area to be 1 implies the assumption that hare density is unaffected by area. However, we can also estimate a coefficient for `log(area)` rather than setting it to 1. For instance, in real life, density may well differ between

large and small areas, perhaps because predator density is also related to area. This hypothesis could be tested by fitting a coefficient for `log(area)` and seeing whether it differs from 1. Adapt the WinBUGS code to achieve this. A Bayesian analysis is extremely suited for this type of question (i.e., to test whether a parameter has a value other than what it is expected to be under a certain hypothesis).
2. *Swiss hare data*: Fit a model that contains both overdispersion, modeled as log-normal random effect, and an offset of `log(area)`, when estimating the difference in mean density between arable and grassland study areas in one year, e.g., 2000 (use a single count per year and area). In a variant of that analysis, estimate the coefficient of `log(area)` to test whether hare density depends on area.