CHAPTER

15

Poisson ANCOVA

OUTLINE

15.1 INTRODUCTION

We may call a Poisson analysis of covariance (ANCOVA) a Poisson regression with both discrete and continuous covariates. In most practical applications, Poisson models will have several covariates and of both types. Therefore, here we look at this important variety of a generalized linear model (GLM). To stress the similarity with the normal linear case, we only slightly alter the inferential setting sketched in Chapter 11. We assume that instead of measuring body mass in asp vipers in three populations in the Pyrenees, Massif Central, and the Jura mountains, leading to a normal model, we had instead assessed ectoparasite load in a dragonfly, the Sombre Goldenring (Fig. 15.1), leading to a Poisson model. We are particularly interested in whether there are more or less little red mites on dragonflies of different size (expressed as wing length) and whether this relationship differs among the three mountain ranges. (Actually, dragonflies don't vary that much in body size, but let's assume there is sufficient variation to make such a study worthwhile.)

Introduction to WinBUGS for Ecologists
DOI: 10.1016/B978-0-12-378605-0.00015-6

FIGURE 15.1 Sombre Goldenring (*Cordulegaster bidentata*), Switzerland, 1995. (*Photo F. Labhardt*)

We will fit the following model to mite count C_i on individual i:

1. Distribution: $C_i \sim \text{Poisson}(\lambda_i)$
2. Link function: log, i.e., $\log(\lambda_i) = \log(E(C_i))$ = linear predictor
3. Lin. predictor: $\log(\lambda_i) = \alpha_{\text{Pyr}} + \beta_1 * x_{\text{MC}} + \beta_2 * x_{\text{Jura}} + \beta_3 * x_{\text{wing}} + \beta_4 * x_{\text{wing}} * x_{\text{MC}} + \beta_5 * x_{\text{wing}} * x_{\text{Jura}}$

Note the great similarity between this model and the one we fitted to the mass of asps in Chapter 11. Apart from the link function, the main other difference is simply that for this model we don't have a dispersion term; the Poisson already comes with a built-in variability. We could model overdispersion in mite counts by using a Poisson-lognormal formulation as in the previous chapter, but we omit such added complexity here.

15.2 DATA GENERATION

We assemble the data set.

```
n.groups <- 3
n.sample <- 100
n <- n.groups * n.sample
```

```
x <- rep(1:n.groups, rep(n.sample, n.groups))    # Population indicator
pop <- factor(x, labels = c("Pyrenees", "Massif Central", "Jura"))

length <- runif(n, 4.5, 7.0)                     # Wing length (cm)
length <- length - mean(length)          # Centre by subtracting the mean
```

We build the design matrix of an interactive combination of length and population:

```
Xmat <- model.matrix(~ pop*length)
print(Xmat, dig = 2)
```

Select the parameter values, i.e., choose values for α_{Pyr}, β_1, β_2, β_3, β_4, β_5

```
beta.vec <- c(-2, 1, 2, 5, -2, -7)
```

Here's the recipe for assembling the mite counts in three steps:

1. we add up all components of the linear model to get the linear predictor, which is the expected mite count on a (natural) log scale,
2. we exponentiate to get the actual value of the expected mite count, and
3. we add Poisson noise.

We again obtain the value of the linear predictor by matrix multiplication of the design matrix (`Xmat`) and the parameter vector (`beta.vec`) (As in the viper example, don't be too harsh on me in case we achieve unnatural levels of parasitation …):

```
lin.pred <- Xmat[,] %*% beta.vec          # Value of lin.predictor
lambda <- exp(lin.pred)                   # Poisson mean
C <- rpois(n = n, lambda = lambda)        # Add Poisson noise

# Inspect what we've created
par(mfrow = c(2,1))
hist(C, col = "grey", breaks = 30, xlab = "Parasite load", main = "", las = 1)
plot(length, C, pch = rep(c("P","M","J"), each=n.sample), las = 1, col =
rep(c("Red","Green","Blue"), each=n.sample), ylab = "Parasite load", xlab = "Wing
length", cex = 1.2)
par(mfrow = c(1,1))
```

We have created a data set where parasite load increases with wing length in the South (Pyrenees, Massif Central) but decreases in the North (Jura mountains); see Fig. 15.2.

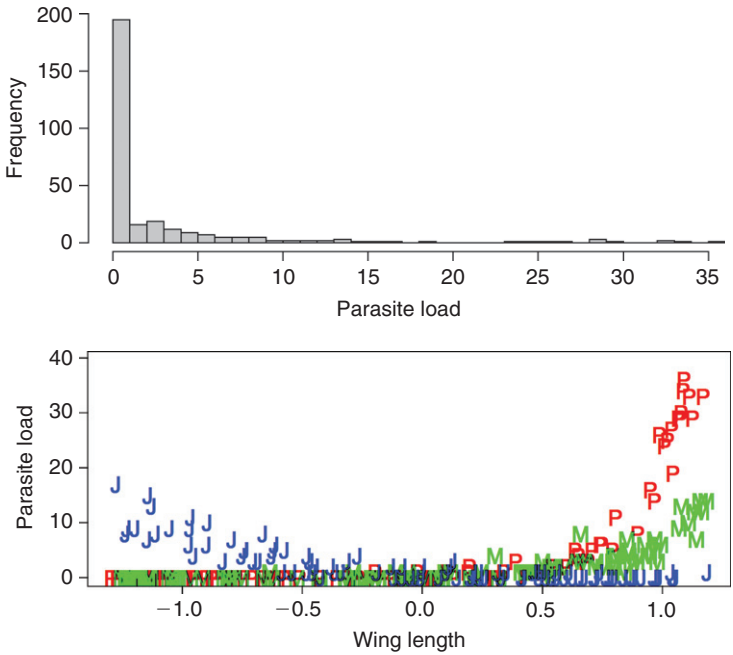

FIGURE 15.2 Top: Frequency distribution of ectoparasite load in Sombre Goldenrings. Bottom: Relationship between parasite load and wing length (deviation from mean, in cm) in three mountain ranges (P – Pyrenees, M – Massif Central, J – Jura mountains).

15.3 ANALYSIS USING R

Again, the R code to fit the model is very parsimonious and the estimates resemble the input values reasonably well (coefficients can be compared directly with `beta.vec`). Obviously, with larger sample sizes the correspondence would be better still (you could try that out, e.g., by setting `n.sample = 1000`).

```
summary(glm(C ~ pop * length, family = poisson))
beta.vec
```

```
> summary(glm(C ~ pop * length, family = poisson))

Call:
glm(formula = C ~ pop * length, family = poisson)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1471  -0.6680  -0.1961   0.2141   2.8843
```

```
Coefficients:
                           Estimate   Std. Error    z value     Pr(>|z|)
(Intercept)                -1.8540       0.2554     -7.260     3.87e-13 ***
popMassif Central           0.6386       0.3463      1.844       0.0652 .
popJura                     1.8046       0.2851      6.330     2.46e-10 ***
length                      4.8199       0.2505     19.244      < 2e-16 ***
popMassif Central:length   -1.6938       0.3516     -4.817     1.46e-06 ***
popJura:length             -6.9059       0.2859    -24.156      < 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: 2402.99  on 299  degrees of freedom
Residual deviance:  205.35  on 294  degrees of freedom
AIC: 684.25

Number of Fisher Scoring iterations: 5

> beta.vec
[1] -2  1  2  5 -2 -7
```

Don't forget that the difference between the coefficients and the beta vector is due to the combined effect of sampling and estimation error. The former means that due to natural variation, 300 dragonflies sampled from large populations can not possibly represent the population values perfectly.

15.4 ANALYSIS USING WinBUGS

15.4.1 Fitting the Model

We simply adapt the code for the normal linear case (Chapter 11) to the Poisson case and again fit a reparameterized model with three separate log-linear regressions.

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

# Priors
 for (i in 1:n.groups){
    alpha[i] ~ dnorm(0, 0.01)       # Intercepts
    beta[i] ~ dnorm(0, 0.01)        # Slopes
 }
```

```
# Likelihood
 for (i in 1:n) {
    C[i] ~ dpois(lambda[i])         # The random variable
    lambda[i] <- exp(alpha[pop[i]] + beta[pop[i]]* length[i])
 }   # Note double-indexing: alpha[pop[i]]

# Derived quantities
# Recover effects relative to baseline level (no. 1)
 a.effe2 <- alpha[2] - alpha[1]      # Intercept Massif Central vs. Pyr.
 a.effe3 <- alpha[3] - alpha[1]      # Intercept Jura vs. Pyr.
 b.effe2 <- beta[2] - beta[1]        # Slope Massif Central vs. Pyr.
 b.effe3 <- beta[3] - beta[1]        # Slope Jura vs. Pyr.

# Custom test
 test1 <- beta[3] - beta[2]            # Slope Jura vs. Massif Central
}
",fill=TRUE)
sink()

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

# Inits function
inits <- function(){list(alpha=rlnorm(n.groups,3,1), beta=rlnorm(n.groups,2,1))}

# Parameters to estimate
params <- c("alpha", "beta", "a.effe2", "a.effe3", "b.effe2", "b.effe3", "test1")

# MCMC settings
ni <- 4500
nb <- 1500
nt <- 5
nc <- 3

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

This is still a simple model that converges fairly rapidly. We inspect the results and compare them with "truth" in the data-generating random process, as well as with the inference from R's function `glm()`.

```
print(out, dig = 3)         # Bayesian analysis
beta.vec                    # Truth in data-generating process
print(glm(C ~ pop * length, family = poisson)$coef, dig = 4) # The ML solution
```

Remember that `alpha[1]` and `beta[1]` in WinBUGS correspond to the intercept and the `length` main effect in the analysis in R and `a.effe2`, `a.effe3`. `b.effe2`, `b.effe3` to the remaining terms of the analysis in R. To ease comparison, these terms are shown in boldface.

```
> print(out, dig = 3)                    # Bayesian analysis
Inference for Bugs model at "glm.txt", fit using WinBUGS,
 3 chains, each with 4500 iterations (first 1500 discarded), n.thin = 5
 n.sims = 1800 iterations saved
             mean    sd   2.5%    25%    50%    75%  97.5%  Rhat n.eff
alpha[1]   -1.844 0.261 -2.384 -2.016 -1.852 -1.666 -1.360 1.017   120
alpha[2]   -1.252 0.233 -1.744 -1.399 -1.235 -1.089 -0.821 1.016   130
alpha[3]   -0.050 0.130 -0.321 -0.136 -0.046  0.037  0.186 1.002  1400
beta[1]     4.808 0.258  4.329  4.628  4.809  4.974  5.329 1.015   140
beta[2]     3.159 0.245  2.712  2.987  3.141  3.322  3.661 1.015   140
beta[3]    -2.087 0.141 -2.367 -2.176 -2.083 -1.987 -1.811 1.002   940
a.effe2     0.592 0.344 -0.063  0.356  0.590  0.825  1.264 1.005   440
a.effe3     1.793 0.291  1.231  1.597  1.793  1.989  2.371 1.015   140
b.effe2    -1.649 0.352 -2.324 -1.887 -1.647 -1.403 -0.970 1.004   530
b.effe3    -6.895 0.296 -7.470 -7.092 -6.891 -6.690 -6.336 1.011   190
test1      -5.246 0.282 -5.823 -5.439 -5.240 -5.044 -4.720 1.015   140
deviance  678.382 3.450 673.597 675.800 677.800 680.200 686.202 1.003  1200

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = Dbar-Dhat)
pD = 6.1 and DIC = 684.5
DIC is an estimate of expected predictive error (lower deviance is better).

> beta.vec# Truth in data-generating process
[1] -2  1  2  5 -2 -7

> print(glm(C ~ pop * length, family = poisson)$coef, dig = 4) # The ML solution
          (Intercept)        popMassif Central              popJura
              -1.8540                   0.6386               1.8046
               length popMassif Central:length       popJura:length
               4.8199                  -1.6938              -6.9059
```

As expected, we find fairly concurrent estimates between the two modes of inference.

15.4.2 Forming Predictions

Finally, let's summarize our main findings from the analysis in a graph. We illustrate the posterior distribution of the relationship between mite load and wing length for each of the three study areas. To do that,

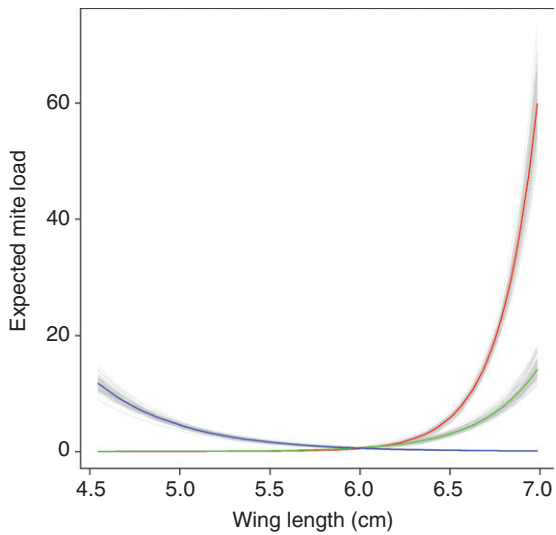

FIGURE 15.3 Predicted relationship between mite load and the size of a dragonfly in three mountain ranges: Pyrenees (red), Massif Central (green), and Jura mountains (blue). Shaded grey lines illustrate the uncertainty in these relationships based on a random sample of 100 from the posterior distribution and colored lines represent posterior means.

we predict mite load for 100 dragonflies in each of the three mountain ranges and plot these estimates along with their uncertainty. We compute the predicted relationship between mite count and wing-length for a sample of 100 of all Markov chain Monte Carlo (MCMC) draws of the involved parameters and plot that (Fig. 15.3).

```
# Create a vector with 100 wing lengths
original.wlength <- sort(runif(100, 4.5, 7.0))
wlength <- original.wlength - 5.75      # 5.75 is approximate mean (correct would be
that from the original data really)

# Create matrices to contain prediction for each winglength and MCMC iteration
sel.sample <- sample(1:1800, size = 100)
mite.load.Pyr <- mite.load.MC <- mite.load.Ju <- array(dim = c(100, 100))

# Fill in these vectors: this is clumsy, but it works
for(i in 1:100) {
   for(j in 1:100)   {
     mite.load.Pyr[i,j] <- exp(out$sims.list$alpha[sel.sample[j],1] +
out$sims.list$beta[sel.sample[j],1] * wlength[i])
      mite.load.MC[i,j] <- exp(out$sims.list$alpha[sel.sample[j],2] +
out$sims.list$beta[sel.sample[j],2] * wlength[i])
      mite.load.Ju[i,j] <- exp(out$sims.list$alpha[sel.sample[j],3] +
out$sims.list$beta[sel.sample[j],3] * wlength[i])
```

```
    }
}

matplot(original.wlength, mite.load.Pyr, col = "grey", type = "l", las = 1, ylab =
"Expected mite load", xlab = "Wing length (cm)")
for(j in 1:100){
    points(original.wlength, mite.load.MC[,j], col = "grey", type = "l")
    points(original.wlength, mite.load.Ju[,j], col = "grey", type = "l")
}
points(original.wlength, exp(out$mean$alpha[1] + out$mean$beta[1] * wlength), col =
"red", type = "l", lwd = 2)
points(original.wlength, exp(out$mean$alpha[2] + out$mean$beta[2] * wlength), col =
"green", type = "l", lwd = 2)
points(original.wlength, exp(out$mean$alpha[3] + out$mean$beta[3] * wlength), col =
"blue", type = "l", lwd = 2)
```

I find this a rather nice plot, but if an editor asks for conventional 95% credible intervals instead, you can give the results of this code:

```
LCB.Pyr <- apply(mite.load.Pyr, 1, quantile, prob=0.025)
UCB.Pyr <- apply(mite.load.Pyr, 1, quantile, prob=0.975)
LCB.MC <- apply(mite.load.MC, 1, quantile, prob=0.025)
UCB.MC <- apply(mite.load.MC, 1, quantile, prob=0.975)
LCB.Ju <- apply(mite.load.Ju, 1, quantile, prob=0.025)
UCB.Ju <- apply(mite.load.Ju, 1, quantile, prob=0.975)

mean.rel <- cbind(exp(out$mean$alpha[1] + out$mean$beta[1] * wlength),
exp(out$mean$alpha[2] + out$mean$beta[2] * wlength), exp(out$mean$alpha[3] +
out$mean$beta[3] * wlength))
covar <- cbind(original.wlength, original.wlength, original.wlength)

matplot(original.wlength, mean.rel, col = c("red", "green", "blue"), type = "l",
lty = 1, lwd = 2, las = 1, ylab = "Expected mite load", xlab = "Wing length (cm)")
points(original.wlength, LCB.Pyr, col = "grey", type = "l", lwd = 1)
points(original.wlength, UCB.Pyr, col = "grey", type = "l", lwd = 1)
points(original.wlength, LCB.MC, col = "grey", type = "l", lwd = 1)
points(original.wlength, UCB.MC, col = "grey", type = "l", lwd = 1)
points(original.wlength, LCB.Ju, col = "grey", type = "l", lwd = 1)
points(original.wlength, UCB.Ju, col = "grey", type = "l", lwd = 1)
```

15.5 SUMMARY

We have generalized the general linear model from the normal to the Poisson case to model the effects on grouped counts of a continuous covariate. The changes involved in doing so in WinBUGS were minor,

and the inclusion of further covariates is straightforward. The Poisson ANCOVA is an important intermediate step for your understanding of the Poisson generalized linear mixed model.

EXERCISES

1. *Multiple Poisson regression*: Invent a new covariate called x_{nonsense}, for instance, and fit this model: $y_i = \alpha_j + \beta_j * x_{\text{wing}} + \delta * x_{\text{nonsense}}$. You can simply create values for x_{nonsense} by sampling a normal or uniform random variable; you don't need to assemble a new data set.
2. *Polynomial Poisson regression*: In addition to the linear effect (on a log-scale) of wing length, check for a quadratic effect also. You may or may not reassemble the data to include that effect.
3. *Swiss hare data*: Take a single count per year and site and fit a Poisson ANCOVA to the counts by expressing counts as a function of site and year. You may ignore the variable site area or incorporate this source of variation by fitting an offset.