

Class 8: Partial pooling and zero-inflation

Andrew Parnell
andrew.parnell@mu.ie



Learning outcomes:

- ▶ Be able to describe the advantages of partial pooling
- ▶ Be able to fit some basic zero inflation and hurdle models
- ▶ Be able to understand and fit some multinomial modelling examples

A false dichotomy: fixed vs random effects

- ▶ We've been fitting a model with varying intercepts and slopes to the earnings data:

$$y_i \sim N(\alpha_{\text{eth}_i} + \beta_{\text{eth}_i} x_i, \sigma^2)$$

where:

$$\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2) \text{ and } \beta_j \sim N(\mu_\beta, \sigma_\beta^2)$$

- ▶ In traditional parlance this is a random effects model
- ▶ When we fit our model we are learning about the values of the slopes and intercepts, and also the values of their means and standard deviations

The extremes of varying vs fixed parameters

- ▶ Now consider what happens when σ_α and σ_β get smaller and smaller. What will happen to the values of the slopes and the intercepts?
- ▶ Alternatively, consider what happens as σ_α and σ_β get larger and larger?
- ▶ Are these still random effects models?

The advantages of borrowing strength

- ▶ The process of σ_α and σ_β getting smaller or larger will control the degree to which the slopes and intercepts are similar to each other
- ▶ If they are similar to each other we say they are *borrowing strength* as data in the other groups is influencing the intercept/slope. This is a powerful idea
- ▶ Mathematically you can write out the estimated mean of the parameters as a weighted average of the group mean and the overall mean where the weights are dependent on the group and overall variance and sample sizes.
- ▶ Because of the weighted nature of the estimate this is often called *partial pooling*

Zero-inflation and hurdle models

- ▶ Let's introduce some new data. This is data from an experiment on whiteflies:

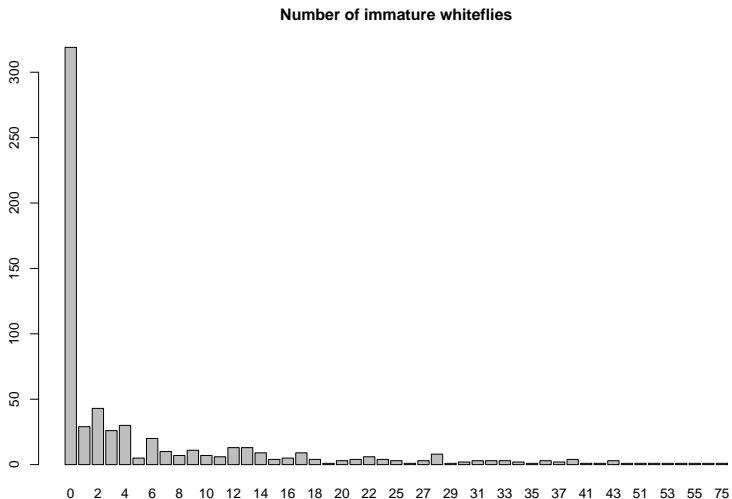
```
wf = read.csv('../data/whitefly.csv')  
head(wf)
```

	##	imm	week	block	trt	n	live	plantid
##	1	15	1	3	5	12	11	1
##	2	16	2	3	5	8	6	1
##	3	28	3	3	5	10	10	1
##	4	17	4	3	5	10	8	1
##	5	9	5	3	5	10	10	1
##	6	28	6	3	5	10	10	1

The response variable here is the count `imm` of immature whiteflies, and the explanatory variables are `block` (plant number), `week`, and treatment `treat`.

Look at those zeros!

```
barplot(table(wf$imm),  
        main = 'Number of immature whiteflies')
```



A first model

- ▶ These are count data so a Poisson distribution is a good start
- ▶ Let's consider a basic Poisson distribution model for Y_i , $i = 1, \dots, N$ observations:

$$Y_i \sim Po(\lambda_i)$$

$$\log(\lambda_i) = \beta_{\text{trt}_i}$$

- ▶ We'll only consider the treatment effect but we could run much more complicated models with e.g. other covariates and interactions

Fitting the model in JAGS

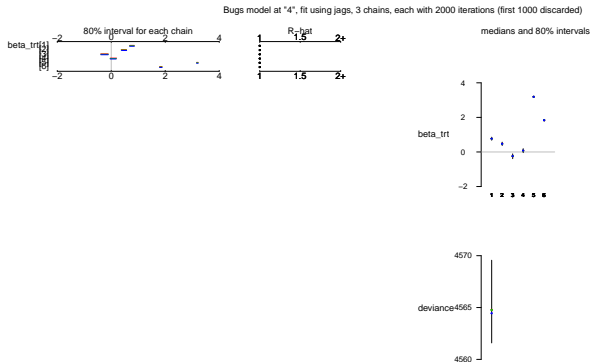
```
model_code = '  
model  
{  
  # Likelihood  
  for (i in 1:N) {  
    y[i] ~ dpois(lambda[i])  
    log(lambda[i]) <- beta_trt[trt[i]]  
  }  
  # Priors  
  for (j in 1:N_trt) {  
    beta_trt[j] ~ dnorm(0, 100^-2)  
  }  
}  
'
```

Running the model

```
jags_run = jags(data = list(N = nrow(wf),  
                             N_trt = length(unique(wf$trt)),  
                             y = wf$imm,  
                             trt = wf$trt),  
                parameters.to.save = 'beta_trt',  
                model.file = textConnection(model_code))
```

Results

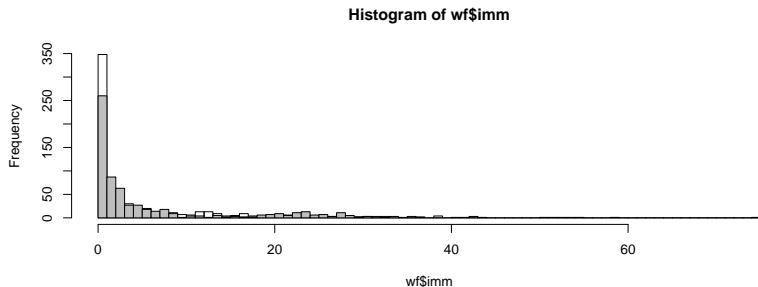
```
plot(jags_run)
```



Some clear treatment effects - treatment 5 in particular

Did the model actually fit well?

```
beta_means = jags_run$BUGSoutput$mean$beta_trt
y_sim_mean = exp(beta_means[wf$trt])
y_sim = rpois(nrow(wf), y_sim_mean)
hist(wf$imm, breaks = seq(0,max(wf$imm)))
hist(y_sim, breaks = seq(0,max(wf$imm)),
      add = TRUE, col = 'gray')
```



What about the zeros?

- ▶ One way of broadening the distribution is through over-dispersion which we have already met:

$$\log(\lambda_i) \sim N(\beta_{\text{trt}_i}, \sigma^2)$$

- ▶ However this doesn't really solve the problem of excess zeros
- ▶ Instead there are a specific class of models called *zero-inflation* models which use a specific probability distribution. The zero-inflated Poisson (ZIP) with ZI parameter q_0 is written as:

$$p(y|\lambda) = \begin{cases} q_0 + (1 - q_0) \times \text{Poisson}(0, \lambda) & \text{if } y = 0 \\ (1 - q_0) \times \text{Poisson}(y, \lambda) & \text{if } y \neq 0 \end{cases}$$

Fitting models with custom probability distributions

- ▶ The Zero-inflated Poisson distribution is not included in Stan or JAGS by default. We have to create it
- ▶ It's possible to create new probability distributions in Stan
- ▶ It's a little bit fiddly to do so in JAGS, we have to use some tricks
- ▶ We will use JAGS to create a mixture of Poisson distributions; A $\text{Poisson}(0)$ distribution for the zeros, and a $\text{Poisson}(\lambda)$ distribution for the rest

Fitting the ZIP in JAGS

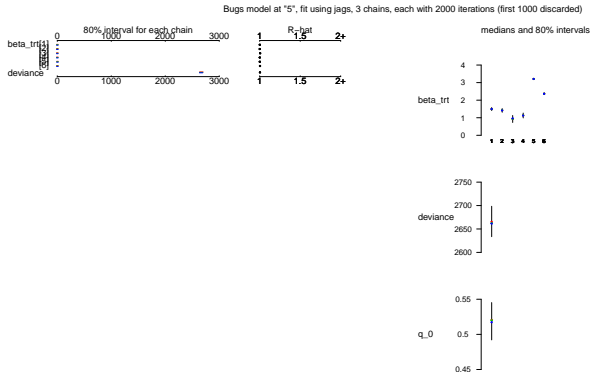
```
model_code = '  
model  
{  
  # Likelihood  
  for (i in 1:N) {  
    y[i] ~ dpois(lambda[i] * z[i] + 0.0001)  
    log(lambda[i]) <- beta_trt[trt[i]]  
    z[i] ~ dbinom(q_0, 1)  
  }  
  # Priors  
  for (j in 1:N_trt) {  
    beta_trt[j] ~ dnorm(0, 100^-2)  
  }  
  q_0 ~ dunif(0, 1)  
}
```

Running the model

```
jags_run = jags(data = list(N = nrow(wf),  
                             N_trt = length(unique(wf$trt)),  
                             y = wf$imm,  
                             trt = wf$trt),  
                parameters.to.save = c('beta_trt', 'q_0'),  
                model.file = textConnection(model_code))
```


Results

```
plot(jags_run)
```

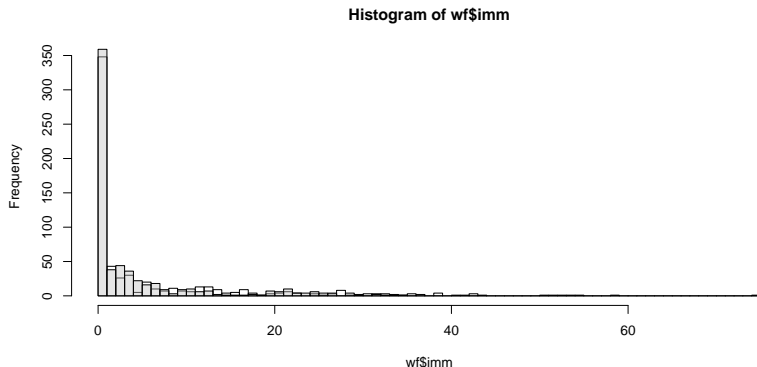


Did it work any better? - code

```
beta_means = jags_run$BUGSoutput$mean$beta_trt
q_0_mean = jags_run$BUGSoutput$mean$q_0[1]
y_sim_mean = exp(beta_means[wf$trt])
rZIP = function(mean, q_0) {
  pois = rpois(length(mean), mean)
  pois[runif(length(mean))<q_0] = 0
  return(pois)
}
y_sim = rZIP(y_sim_mean, q_0_mean)
```

Did it work any better? - picture

```
hist(wf$imm, breaks = seq(0,max(wf$imm)))  
hist(y_sim, breaks = seq(0,max(wf$imm)),  
     add = TRUE, col = rgb(0.75,0.75,0.75,0.4))
```



Some more notes on Zero-inflated Poisson

- ▶ This model seems to predict the number of zeros pretty well. It would also be interesting to perhaps try having a different probability of zeros (q_0) for different treatments
- ▶ It might be that the other covariates explain some of the zero behaviour
- ▶ We could further add in both zero-inflation and over-dispersion

An alternative: hurdle models

- ▶ ZI models work by having a parameter (here q_0) which is the probability of getting a zero, and so the probability of getting a Poisson value (which could also be a zero) is 1 minus this value
- ▶ An alternative (which is slightly more complicated) is a hurdle model where q_0 represents the probability of the *only* way of getting a zero. With probability $(1-q_0)$ we end up with a special Poisson random variable which has to take values 1 or more
- ▶ In some ways this is richer than a ZI model since zeros can be deflated or inflated
- ▶ This is a bit fiddlier to fit in JAGS

A hurdle-Poisson model in JAGS

```
model_code = '  
model  
{  
  # Likelihood  
  for (i in 1:N) {  
    y[i] ~ dpois(lambda[i])T(1,)  
    log(lambda[i]) <- beta_trt[trt[i]]  
  }  
  for(i in 1:N_0) {  
    y_0[i] ~ dbin(q_0, 1)  
  }  
  # Priors  
  for (j in 1:N_trt) {  
    beta_trt[j] ~ dnorm(0, 100^-2)  
  }  
  q_0 ~ dunif(0, 1)  
}
```

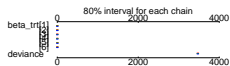
Running the model

```
jags_run = jags(data = list(N = nrow(wf[wf$imm > 0,]),  
                             N_trt = length(unique(wf$trt)),  
                             y = wf$imm[wf$imm > 0],  
                             y_0 = as.integer(wf$imm == 0),  
                             N_0 = nrow(wf),  
                             trt = wf$trt[wf$imm > 0]),  
parameters.to.save = c('beta_trt', 'q_0'),  
model.file = textConnection(model_code))
```

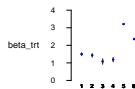
Results

```
plot(jags_run)
```

Bugs model at "6", fit using jags, 3 chains, each with 2000 iterations (first 1000 discarded)



medians and 80% intervals



Some final notes on ZI models

- ▶ To complete the Poisson-Hurdle fit we would need to simulate from a truncated Poisson model. This starts to get very fiddly though - see the `jags_examples` repository for worked examples
- ▶ We can extend these models further by using a better count distribution such as the negative binomial which has an extra over-dispersion parameter
- ▶ We can also add covariates into the zero-inflation component, though it is not always clear whether this is desirable

The multinomial distribution

- ▶ Multinomial data can be thought of as multivariate discrete data
- ▶ It's usually used in two different scenarios:
 1. For classification, when you have an observation falling into a single one of K possible categories
 2. For multinomial regression, where you have a set of counts which sum to a known value N
- ▶ We will just consider the multinomial regression case, whereby we have observations $y_i = [y_{i1}, \dots, y_{iK}]$ where the sum $\sum_{k=1}^K y_{ik} = N_i$ is fixed
- ▶ The classification version is a simplification of the regression version

Some new data! - pollen

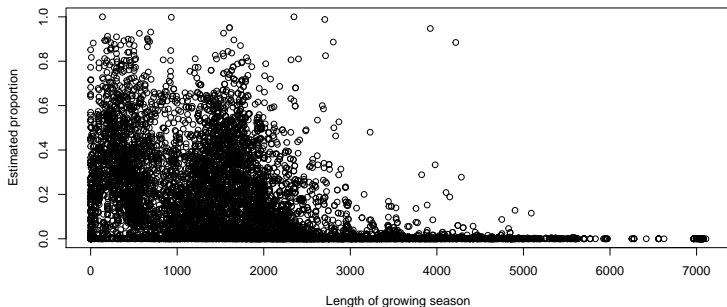
```
pollen = read.csv('../data/pollen.csv')  
head(pollen)
```

##	GDD5	MTCO	Abies	Alnus	Betula	Picea	Pinus.D	Quercus.D	Gramineae
## 1	1874	-7.9	0	50	158	7	721	22	0
## 2	1623	-5.5	0	38	28	302	537	19	0
## 3	1475	-4.7	0	276	183	110	136	0	0
## 4	1360	-8.8	0	111	354	141	364	0	0
## 5	1295	-6.9	0	91	50	151	708	0	0
## 6	1539	-7.8	0	51	194	82	673	0	0

These data are pollen counts of 7 varieties of pollen from modern samples with two covariates

Some plots

- ▶ The two covariates represent the length of the growing season (GDD5) and harshness of the winter (MTCO)
- ▶ The task is to find which climate regimes each pollen variety favours



A multinomial model

- ▶ The multinomial distribution is often written as:

$$[y_{i1}, \dots, y_{iK}] \sim \text{Mult}(S_i, \{p_{i1}, \dots, p_{iK}\})$$

or, for short:

$$y_i \sim \text{Mult}(S_i, p_i)$$

- ▶ The key parameters here are the probability vectors p_i . It's these we want to use a link function on to include the covariates
- ▶ We need to be careful as each must sum to one: $\sum_{k=1}^K p_{ik} = 1$. Any link function must satisfy this constraint

Prior distributions on probability vectors

- ▶ When $K = 2$ we're back the binomial-logit we met in the first day, and we can use the logit link function
- ▶ When $K > 2$ a common function to use is the *soft-max* function:

$$p_{ik} = \frac{\exp(z_{ik})}{\sum_{j=1}^K \exp(z_{ij})}$$

- ▶ This is a generalisation of the logit function
- ▶ The next layer of our model sets, e.g.:

$$z_{ik} = \beta_0 + \beta_1 \text{GDD5}_i + \gamma_2 \text{MTCO}_i + \dots$$

JAGS code

```
model_code = '  
model  
{  
  # Likelihood  
  for (i in 1:N) { # Observaton loops  
    y[i,] ~ dmulti(p[i,], S[i])  
    for(j in 1:M) { # Category loop  
      exp_z[i,j] <- exp(z[i,j])  
      p[i,j] <- exp_z[i,j]/sum(exp_z[i,])  
      z[i,j] <- beta[j,]%*%x[i,]  
    }  
  }  
  # Prior  
  for(j in 1:M) {  
    for(k in 1:K) {  
      beta[j,k] ~ dnorm(0, 0.1^-2)  
    }  
  }  
}
```

Let's fit it (first 500 obs only)

```
model_data = list(N = nrow(pollen[1:500,]),
                  y = pollen[1:500,3:9],
                  x = cbind(1, scale(cbind(pollen[1:500,1:2],
                                           pollen[1:500,1:2])),
                           0, 1),
                  S = pollen[1:500,10],
                  K = 5, # Number of covars
                  M = 7) # Number of categories

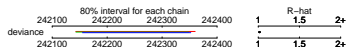
# Run the model
model_run = jags(data = model_data,
                 parameters.to.save = c("p"),
                 model.file = textConnection(model_code))
```

```
## Compiling model graph
##   Resolving undeclared variables
##   Allocating nodes
## Graph information:
##   Observed stochastic nodes: 500
##   Unobserved stochastic nodes: 35
##   Total graph size: 15204
```

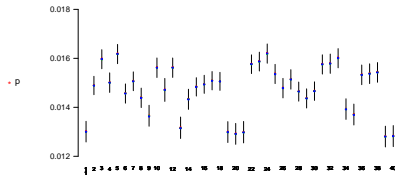
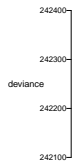

Results 1

```
plot(model_run)
```

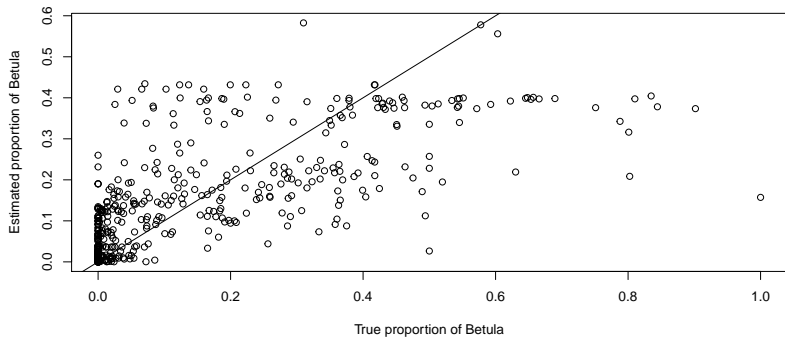
Bugs model at "7", fit using jags, 3 chains, each with 2000 iterations (first 1000 discarded)



medians and 80% intervals



Results 2



Notes about this model

- ▶ This model is not going to fit very well, since it is unlikely that a linear relationship between the covariates and the pollen counts will match the data
- ▶ It might be better to use e.g. a spline model (not covered in this course, but we can talk about it)
- ▶ Similarly we might need some complex interactions between the covariates as they are strongly linked
- ▶ We have constrained the parameters here so that the slopes and intercepts borrow strength across species. Does this make sense? What else could we do?

Some final notes about multinomial models

- ▶ These models can be a pain to deal with as there are tricky constraints on the β parameters to make them all sum to 1. Instead it's often easier to just put a tight prior distribution on them, e.g. $\beta \sim N(0, 0.1)$
- ▶ The `softmax` function is one choice but there are lots of others (logistic ratios, the Dirichlet distribution, ...)
- ▶ Whilst the classification version of this model just has binary y_i (with just a single 1 in it, i.e. $S_i = 1$) most packages (including JAGS and Stan) have a special distribution (e.g. `dcat` in JAGS) for this situation

Summary

- ▶ We have seen how partial pooling is a balance between a model of complete independence and complete dependence between groups
- ▶ We have fitted some zero inflated and hurdle Poisson models in JAGS
- ▶ We have seen a simple multinomial logistic regression