

Bayesian Analysis of Multinomial N-mixture models in BUGS

Multinomial N-mixture in BUGS

- Bayesian analysis of multinomial observation models does not pose any novel technical difficulty.
- WinBUGS JAGS (and other BUGS) have a multinomial distribution function.
- In principle we just specify the multinomial/Poisson model directly:

```
y[i,]<-c(y[i,],NA) # missing value for "not captured"
```

```
# Then, in WinBUGS, do this:
```

```
y[i,] ~ dmulti( probs[i,], N[i] )
```

```
N[i] ~ dpois(lambda[i])
```

Multinomial N-mixture in BUGS

```
y[i,]<-c(y[i,],NA)    # missing value for "not captured"  
  
# Then, in WinBUGS, do this:  
y[i,] ~ dmulti( probs[i,], N[i] )  
N[i] ~ dpois(lambda[i])
```

However, this construction doesn't work (in WinBUGS). Cannot have “random” sample size in multinomial distribution. (not so Binomial!)

Approaches to Bayesian analysis

We have 3 options for analysis in BUGS:

- (1) Multinomial/Poisson mixture has Poisson marginals. Use the Poisson marginal. (this is too easy!).
- (2) Can use a “data augmentation” trick (Converse and Royle 2012) with individual-level encounter histories.
- (3) Can express the model in terms of the conditional multinomial observation model. i.e., condition on n_i = number of individuals captured at site i -- the “**3 part model**”.

Topics in Bayesian analysis

- 3-part model
- Goodness-of-fit
- Model selection
- Poisson model
- Poisson model with random effects
- Data Augmentation

The 3-part (conditional multinomial) model

Statistically equivalent formulation of the multinomial N-mixture – a reparameterization:

1. $\mathbf{y}_i | n_i \sim \text{Multinomial}(n_i, \boldsymbol{\pi}_i^c)$
2. $n_i \sim \text{Binomial}(N_i, 1 - \pi_0)$
3. $N_i \sim \text{Poisson}(\lambda_i)$

where

$$\boldsymbol{\pi}_i^c = \frac{\boldsymbol{\pi}_h}{1 - \pi_0}$$

```

model {
# Prior distributions
p0 ~ dunif(0,1)
alpha0 <- logit(p0)
alpha1 ~ dnorm(0, 0.01)
beta0 ~ dnorm(0, 0.01)
beta1 ~ dnorm(0, 0.01)
beta2 ~ dnorm(0, 0.01)
beta3 ~ dnorm(0, 0.01)

for(i in 1:M){ # Loop over sites
  # Conditional multinomial cell probabilities
  pi[i,1] <- p[i]
  pi[i,2] <- p[i]*(1-p[i])
  pi[i,3] <- p[i]*(1-p[i])*(1-p[i])
  pi[i,4] <- p[i]*(1-p[i])*(1-p[i])*(1-p[i])
  pi0[i] <- 1 - (pi[i,1] + pi[i,2] + pi[i,3] + pi[i,4])
  pcap[i] <- 1 - pi0[i]
  for(j in 1:4){
    pic[i,j] <- pi[i,j] / pcap[i]
  }
}

# logit-linear model for detection: understory cover effect
logit(p[i]) <- alpha0 + alpha1 * X[i,1]
}

# Model specification, three parts:
y[i,1:4] ~ dmulti(pic[i,1:4], n[i]) # component 1 uses the
# conditional cell probabilities
n[i] ~ dbin(pcap[i], N[i]) # component 2 is a model
# for the observed sample size
N[i] ~ dpois(lambda[i]) # part 3 is the process model

# log-linear model for abundance: UFC + TRBA + UFC:TRBA
log(lambda[i])<- beta0 + beta1*X[i,1] + beta2*X[i,2] +
beta3*X[i,2]*X[i,1]
}
}

```

Have to write out
the cell
probabilities
explicitly

Goodness-of-fit Using Bayesian p-values

- It is natural to think of “overall fit” as having two components: How well does the encounter model fit? How well does the abundance model fit?
- Can we evaluate them independently? The 3-part model leads to a natural formulation of this dual fit assessment strategy.
 - Fit of multinomial model conditional on n_i .
 - Fit of the model for n – should be sensitive to wrong model for because $E(n_i) = \lambda_i p_{cap}$ contains the variation in N_i (and spatial variation in p)

Implementation of 2-part GoF idea

```
for(i in 1:nsites){

  ncap.fit[i] ~ dbin(pcap[i],N[i])
  y.fit[i,1:4] ~ dmulti(muc[i,1:4],ncap[i])
  for(t in 1:4){
    e1[i,t]<- muc[i,t]*ncap[i] # Expected value
    resid1[i,t]<- pow(pow(y[i,t],0.5)-pow(e1[i,t],0.5),2)
    resid1.fit[i,t]<- pow(pow(y.fit[i,t],0.5) - pow(e1[i,t],0.5),2)
  }
  e2[i]<- pcap[i]*lambda[i] # Expected value
  resid2[i]<- pow( pow(ncap[i],0.5) - pow(e2[i],0.5),2)
  resid2.fit[i]<- pow( pow(ncap.fit[i],0.5) - pow(e2[i],0.5),2)
}

fit1.data<- sum(resid1[,])
fit1.post<- sum(resid1.fit[,])
fit2.data<- sum(resid2[])
fit2.post<- sum(resid2.fit[])
```

Goodness-of-fit Using Bayesian p-values

```
> mean(out$sims.list$fit1.post>out$sims.list$fit1.data)
[1] 0.7076667
```

```
> mean(out$sims.list$fit2.post>out$sims.list$fit2.data)
[1] 0.4556667
```

No lack of fit is indicated...?????

Goodness-of-fit: Research Question

The power of any particular fit statistic to any particular departure from the model is unknown and no studies have been published.

Model Selection in BUGS: Computing posterior model probabilities

Basic idea: Expand model to include a set of binary indicator variables $w_k = 1$ if variable k is in the model (Kuo and Mallick 1998)

Model selection \equiv estimating $\Pr(w_k = 1)$. (R&D Book, sec. 3.4.3)

Expanded linear predictor:

$$\log(\lambda_i) = \beta_0 + w_1\beta_1x_{i1} + w_2\beta_2x_{i2} + w_1w_2w_3x_{i1}x_{i2}$$

$$w_1 \sim \text{Bern}(0.5)$$

$$w_2 \sim \text{Bern}(0.5)$$

$$w_3 \sim \text{Bern}(0.5)$$

Models are characterized by the sequence $(w_1, w_2, w_1w_2w_3)$

- Estimate functions of w_k e.g., $\Pr(w_k = 1)$
- Sensitivity to prior. Posterior model probabilities are sensitive to choice of prior distribution on β . See Link and Barker (2010).

Model Selection in BUGS: Computing posterior model probabilities

```
model {
  beta0 ~ dnorm(0, .1)
  Beta1 ~ dnorm(0, .1)
  beta2 ~ dnorm(0, .1)
  beta3 ~ dnorm(0, .1)
  w1 ~ dbern(.5)
  w2 ~ dbern(.5)
  w3 ~ dbern(.5)

  p0~dunif(0,1)

  for(i in 1:nsites){
    p[i]<- p0 # could have covariates here
    mu[i,1] <- p[i]
    mu[i,2] <- p[i]*(1-p[i])
    mu[i,3] <- p[i]*(1-p[i])*(1-p[i])
    mu[i,4] <- p[i]*(1-p[i])*(1-p[i])*(1-p[i])
    pi0[i]<- 1 - mu[i,1]-mu[i,2]-mu[i,3]-mu[i,4]
    pcap[i]<-1-pi0[i]

    for(j in 1:4){
      muc[i,j] <- mu[i,j]/pcap[i]
    }
    y[i,1:4] ~ dmulti(muc[i,1:4],ncap[i])
    ncap[i] ~ dbin(pcap[i],N[i])
    N[i] ~ dpois(lambda[i])
    log(lambda[i])<- beta0 + w1*beta1*X[i,1] + w2*beta2*X[i,2] + w1*w2*w3*beta3*X[i,2]*X[i,1]
  }
}
```

Model Selection in BUGS: Computing posterior model probabilities

Post-processing to obtain model frequencies -- combine the unique values of (w_1, w_2, w_3) into distinct models. i.e., $(1,0,0)$, $(0,1,0)$, $(1,1,0)$, etc.. Note: When w_3 represents an interaction we want to use $(w_1, w_2, w_1w_2w_3)$ so that the model has the interaction only if the main effects are present.

```
w1<-out$sims.list$w1
w2<-out$sims.list$w2
# new "w3" =1 only if the interaction is in the
#   model, means w1 = 1 AND w2=1
w3<-out$sims.list$w3 * w1 * w2
mod<-paste(w1,w2,w3)
```

Sensitivity to prior distributions

```
Prior: beta ~ dnorm(0, .1)
```

```
> table(mod)
```

```
mod
```

0 0 0	0 1 0	1 0 0	1 1 0	1 1 1
2176	517	300	6	1

```
Prior: beta ~ dnorm(0, .01)
```

0 0 0	0 1 0	1 0 0	1 1 0
2760	154	78	8

```
Prior: beta ~ dnorm(0, .2)
```

0 0 0	0 1 0	1 0 0	1 1 0	1 1 1
2006	637	340	16	1

Model selection summary

Model selection based on posterior model probabilities, when models represent different fixed covariates, is easy to accomplish using variable weights.

The prior makes a difference, so be careful.

Poisson formulation of the model

If $N_i \sim \text{Poisson}(\lambda_i)$ then the marginal distribution of the data is also Poisson!

y_{ih} = frequency of encounter history h at site i

$$y_{ih} \sim \text{Poisson}(\pi_{ih}\lambda_i)$$

i.e., model is just a Poisson GLM!

```

model {
  # Prior distributions
  p0 ~ dunif(0,1)
  alpha0 <- logit(p0)
  alpha1 ~ dnorm(0, 0.01)
  beta0 ~ dnorm(0, 0.01)
  beta1 ~ dnorm(0, 0.01)
  beta2 ~ dnorm(0, 0.01)
  beta3 ~ dnorm(0, 0.01)

  for(i in 1:M){
    # logit-linear model for detection: understory cover effect
    logit(p[i]) <- alpha0 + alpha1 * X[i,1]
    # log-linear model for abundance: UFC + TRBA + UFC:TRBA
    log(lambda[i])<- beta0 + beta1*X[i,1] + beta2*X[i,2] + beta3*X[i,2]*X[i,1]

    # Poisson parameter = multinomial cellprobs x expected abundance
    pi[i,1] <- p[i] * lambda[i]
    pi[i,2] <- p[i] * (1-p[i]) * lambda[i]
    pi[i,3] <- p[i] * (1-p[i]) * (1-p[i]) * lambda[i]
    pi[i,4] <- p[i] * (1-p[i]) * (1-p[i]) * (1-p[i]) * lambda[i]

    for(j in 1:4){
      y[i,j] ~ dpois(pi[i,j])
    }
    # Generate predictions of N[i]
    N[i] ~ dpois(lambda[i])
  }
}

```

Poisson model with random effects

- [see R script]

Data Augmentation (DA)

Motivation

- ovenbird, ALFL, MHB data are all classical “capture-recapture” data/models but for those we formulated the model in terms of **encounter frequencies** for each site (a multinomial vector that is site specific). We modeled the latent N_i variables as Poisson (or NB, etc..)
- In those models there is no **INDIVIDUAL IDENTITY** (only a site identity)
- DA gives us an alternative formulation of the model that preserves individual identity so that we may model individual effects in addition to site effects

Data Augmentation (DA)

Conceptual approach

- The idea of DA is to stack all of the site-specific individual encounter history data sets into one large data set (the “stacked data set”) and treat the data set as a single capture-recapture data set. Now N is the population size of the “pooled population”. That is, the population size among all sampled sites.

```
y <- as.matrix(alfl[,c("interval1", "interval2", "interval3")] )
head(y)
      interval1 interval2 interval3
[1,]          1          1          1 # Each row = individual
[2,]          1          0          1 #   ALL SITES POOLED
[3,]          0          1          1
[4,]          1          1          1
[5,]          0          1          1
[6,]          1          0          1
site <- as.numeric(alfl$id)
head(site)
[1] 1 1 2 2 2 2
```

Data Augmentation (DA)

Conceptual approach

- What is the model for the “stacked” data set?
- Main technical challenge: N is unknown! DA is meant primarily to deal with the unknown N problem.
- First we switch topics and talk about analyzing “basic” capture-recapture models using DA to make the core methodological idea clear.

Sampling a closed population

A typical closed population sampling data set:

Ind.	- occasion -				
1	0	1	0	1	1
2	0	0	1	0	0
3	1	1	0	0	0
4	0	0	1	1	0
5	0	1	1	1	1
6	0	0	1	1	0
7	1	1	1	1	1
8	1	0	1	1	0

Here we sampled a population of size N repeatedly ($J=5$ times) and observed $n=8$ individuals. We wish to estimate N . How do we do that?

Models M_0 , M_h , M_t , M_b , M_{bh} , M_{th} , M_{bth} , Individual covariate models M_x , etc..

This is just one "Site" (sampling of one population)

Bayesian analysis of closed capture-recapture models: The basic problem of variable dimension data/parameters

If N is known, CR model is just a logistic regression:

```
model {  
  
p~dunif(0,1)  
  
for (i in 1:N) {  
  for( j in 1:J) {  
    y[i,j]~dbern(p)  
  }  
}  
  
}
```

But N is not known. Conceptually we could just put a prior on N , e.g., $N \sim \text{Dunif}(0, 1000)$, and analyze the model using standard methods of MCMC

However, the size of the data set, N , is a parameter of the model so as N is updated in the MCMC algorithm **the size of the data set must change. Can't do this in WinBUGS/JAGS.**

Bayesian analysis of closed population models

- Prior distributions:
 - $N \sim \text{Dunif}(0, M)$, for M some big number (Fixed)
 - $p \sim \text{uniform}(0,1)$
- Not amenable to a naïve implementation by MCMC (esp in BUGs/JAGS) because N , a parameter, which is the size of the data set, which has to be fixed! Also “variable dimension parameter space”
 - Therefore:
 - RJMCMC/“Trans-dimensional” Gibbs sampling
 - Data augmentation <- easier, can be done in BUGS

Data augmentation: Heuristic

- $N \sim \text{Dunif}(0, M)$ implies a “data set” with $M-n$ all-zero encounter histories. Some of the $y=0$ observations correspond to real individuals and some of them do not.
 - Same as:
 - $N|\psi \sim \text{Bin}(M, \psi)$ **## KEY POINT!**
 - $\psi \sim \text{uniform}(0, 1)$
- Implementation: We add too many zeroes to the dataset – creating a zero-inflated version of the known- N dataset
- Model for the augmented data set is a zero-inflated binomial
- **THIS IS AN OCCUPANCY MODEL!**

Heuristic development

Occupancy data

Site | - occasion - |

1	0	1	0	1	1
2	0	0	1	0	0
3	1	1	0	0	0
4	0	0	1	1	0
5	0	1	1	1	1
6	0	0	1	1	0
7	1	1	1	1	1
8	1	0	1	1	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
M	0	0	0	0	0

Zeros are observed. Allocate zeros to “fixed” and “sampling”

Closed pop. sampling

Ind. | - occasion - |

1	0	1	0	1	1
2	0	0	1	0	0
3	1	1	0	0	0
4	0	0	1	1	0
5	0	1	1	1	1
6	0	0	1	1	0
7	1	1	1	1	1
8	1	0	1	1	0

Zeros are NOT observed. How many “sampling” zeros are there?

Closed pop. + DA

Ind. | - occasion - |

1	0	1	0	1	1
2	0	0	1	0	0
3	1	1	0	0	0
4	0	0	1	1	0
5	0	1	1	1	1
6	0	0	1	1	0
7	1	1	1	1	1
8	1	0	1	1	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
M	0	0	0	0	0

Bound $N \leq M$ where M is fixed.

Treat Model M0 as an occupancy model.

DA and occupancy models

- DA makes capture-recapture models the same as occupancy models.
- The parameter ψ replaces population size N . They are related as follows: $N \sim \text{Binomial}(M, \psi)$
- Occupancy model is implemented at the individual level by introducing latent occupancy state $z_i \sim \text{Bern}(\psi)$

Why can we do this?

- $N \sim \text{Unif}(0, M)$
- **Same as:**
 - $N | \psi \sim \text{Bin}(M, \psi)$ $M = \text{fixed}$
 - $\psi \sim \text{uniform}(0, 1)$

This 2-part prior implies: $N \sim \text{Uniform}(0, M)$, standard distribution theory result
- **Same as:**
 - $z[i] \sim \text{Bern}(\psi)$ for $i=1, 2, \dots, M$ “data augmentation variables”
 - $y[i] \sim \text{Bern}(p * z[i])$
 - $\psi \sim \text{dunif}(0, 1)$ “data augmentation parameter”
- **The augmented data create a super-population of individuals available to be “recruited” by the MCMC algorithm.**

Fit model M0 in BUGS/JAGS using DA

2 formulations of Model M0 in BUGS:

Encounter frequencies

```
model {  
  psi~dunif(0, 1)  
  p~dunif(0,1)  
  for (i in 1:M){  
    z[i]~dbern(psi)  
    tmp[i]<-p*z[i]  
    y[i]~dbin(tmp[i],K)  
  }  
  N<-sum(z[1:M])  
}
```

Binary encounter events

```
model {  
  psi~dunif(0, 1)  
  p~dunif(0,1)  
  for (i in 1:M){  
    z[i]~dbern(psi)  
    for(k in 1:K){  
      tmp[i,k]<-p*z[i]  
      y[i,k]~dbin(tmp[i,k],1)  
    }  
  }  
  N<-sum(z[1:M])  
}
```

Data Augmentation (DA)

Analysis of ALFL data using DA

```
y <- as.matrix(alfl[,c("interval1", "interval2", "interval3")] )
head(y)
      interval1 interval2 interval3
[1,]          1          1          1 # Each row = individual
[2,]          1          0          1 #   ALL SITES POOLED
[3,]          0          1          1
[4,]          1          1          1
[5,]          0          1          1
[6,]          1          0          1
site <- as.numeric(alfl$id)
head(site)
[1] 1 1 2 2 2 2
```

Work session

- Analysis of the ALFL data by data augmentation

DA for site-structured models

- Change of notation!!!
- If we have data classified by **both site and individual**... we need a new indexing scheme
 - i = individual (not site)
 - s = site

DA for site-structured models

- **DA**: The key idea of DA is to preserve an individual-level formulation of capture-recapture models which can be analyzed easily by MCMC (i.e., in BUGS).
- DA: we analyze the “stacked” data set. Take the data set from each site and pile them up on top of each other.
- We also have **site-structured data**.
 - Introduce an individual covariate $g[i]$ (g for “group”) which determines the site membership of each individual i
 - In BUGS: $g[i] \sim dcat(probs[])$

DA for site-structured models

- We also have **site-structured data**.
 - Introduce an individual covariate $g[i]$ (g for “group”) which determines the site membership of each individual i
 - In BUGS: $g[i] \sim dcat(probs[])$
- When we use DA to analyze models the “site membership” of individuals appears as a **categorical individual covariate**

DA for site-structured models

- Individual *site covariate*: $g[i] \sim \text{dcat}(\text{probs}[i])$
- What is $\text{probs}[i]$????
- Derives from the assumption for N_s

DA for site-structured models

1. Group membership has a categorical distribution $g_i \sim \text{Categorical}(\boldsymbol{\pi})$ with cell probabilities

$$\pi_s = \frac{\lambda_s}{\sum_s \lambda_s}$$

where λ_s is the mean of the Poisson abundance just as in an ordinary multinomial or N -mixture model.

2. The data augmentation parameter ψ is derived from the λ_s parameters in the following way:

$$\psi = \frac{\sum_s \lambda_s}{M}$$

The end