Hierarchical Modeling Introduction

Andrew Crosby

06 March, 2018

The first step is to download the JAGS (Just Another Gibbs Sampler) program here: <https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Windows/> . Make sure you’ve got the right version, i.e. it is compatable with your version of R (read the note at the bottom of the page). Then, make sure you have the R2jags package for R. There are several other packages for running JAGS from R, but we will stick with this one for now.

JAGS is similar to WinBUGS and uses the same language to write the models, but is much faster and less buggy[*sic*]. Both programs implement what’s called a Gibbs Sampler, which is a specific alogrithm for doing the MCMC sampling in Bayesian analysis. JAGS is a litte bit pickier about initial values but that is easy to get around.

### The Most Basic Occupancy Model

We will begin with a basic occupancy model. Following Kéry (2010) we will simulate data to reflect the occupancy and detection processes and then analyize it using a few different versions of a hierarhical model. The basic model reflects a simple process, where the realized occupancy state *z* at each site *i* is drawn from a bernoulli distibution with probability :

and detecion at site *i* during survey *j*, conditional on occupancy, is also the result of a Bernouli process with probability *p*:

We will start of with a simple exercise where we consider a situation where occupancy and detection probability are both assumed constant across the study area. This isn’t realistic, but it serves to illustrate our process.

set.seed(2) # Set the seed so that the numbers come out the same each time  
R <- 100 # The number of sites  
T <- 3 # The number of surveys  
  
psi <- 0.7 # The probability of occupancy at each site  
p <- 0.4 # The probability of detection during each survey  
  
# Simulate the data using the above parameters  
# --------------------------------------------  
  
# Simulate the occupancy data  
presence <- rbinom(n = R, size = 1, prob = psi) # This creates a vector of 1's and 0's where the probability of being a 1 is psi  
sum(presence)/100 # The true occuancy rate across all sites. It is slightly different from the occupancy probability because it is the result of a random process with probabiltiy psi.

## [1] 0.69

# Simulate the detection process  
y <- matrix(NA, nrow = R, ncol = T)  
for(i in 1:R){  
 y[i, ] <- rbinom(n = T, size = 1, prob = p\*presence[i])  
}  
  
naive.est <- sum(apply(y, 1, max))/100 # The naive estimate of occupancy (without accounting for detection)  
naive.est

## [1] 0.57

We see from the above numbers that the true occupancy rate is 0.69 while the naïve estimate (without correcting for detection) is only 0.57.

To analyze the above data, we will use what can be called a “model of the mean”, because occupancy is assumed constant so the model is estimating the mean occupancy probability across all sites rather than individual site-level probabilities.

In the BUGS language, the model looks like this:

sink("basic\_model.txt")  
cat("  
model{  
 # The prior disturbutions on psi and p  
 psi ~ dunif(0, 1)  
   
 p ~ dunif(0, 1)  
   
 # The likelihood  
 for(i in 1:R){  
 # Model 1: the occupancy process  
 z[i] ~ dbern(psi)  
   
 # Model 2: the detection process, conditional on occupancy  
 for(j in 1:T){  
 y[i, j] ~ dbern(z[i]\*p)  
 }  
 }  
}  
", fill = TRUE)  
sink()

I want to pasue and make a point here. The reason for simulating data is that it reflects our ideas about the data-generating process. The thing to notice about the model above is how it matches very closely to the simulation code. This is becasue the model itself reflects our assumptions about the process that generated the data. We generated the occcupancy data as 100 random draws from a binomial distribution, with a single trial at each replicate (representing a single coin flip, so a bernoulli trial), with probability of success . In the model description, this is represented as . In the model code above, this is specified under Model 1 as “z[i] ~ dbern(psi).” This concept that the model reflects the data-generating process is very important when learning to build Bayesian models.

# Compile the data  
data <- list(R = R, T = T, y = y)  
  
# Set the parameters to monitor  
params <- c("psi", "p")  
  
# Specify the initial values - this is critical in JAGS, and is why we use the naive occupancy as initial values for z  
zinit <- apply(y, 1, max)  
inits <- function(){list(z = zinit, psi = runif(1, 0, 1), p = runif(1, 0, 1))}  
  
nc <- 3 # The number of mcmc chains to run  
ni <- 1000 # The total number of iterations (i.e. the number of random draws from the posterior distribution)  
nb <- 100 # The number of iterations used for burn-in (like a model warm-up)  
nt <- 5 # The thinning rate, meaning that the output will only save 1 out of every 5 iterations. This can save a great deal of computer memory for large models  
  
# Load the R2jags package so that R can talk to JAGS  
library(R2jags)

out <- jags(data, inits, params, model.file = "basic\_model.txt", n.chains = nc, n.iter = ni, n.burnin = nb, n.thin = nt)  
print(out, digits = 2)

## module glm loaded

## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 300  
## Unobserved stochastic nodes: 102  
## Total graph size: 508  
##   
## Initializing model

## Inference for Bugs model at "C:/Users/crosby/AppData/Local/Temp/RtmpMfoT1f/model1cdc729622a7.txt", fit using jags,  
## 3 chains, each with 1000 iterations (first 100 discarded), n.thin = 5  
## n.sims = 540 iterations saved  
## mu.vect sd.vect 2.5% 25% 50% 75% 97.5% Rhat n.eff  
## p 0.35 0.05 0.27 0.32 0.35 0.38 0.45 1.01 240  
## psi 0.78 0.09 0.62 0.72 0.78 0.84 0.96 1.00 540  
## deviance 307.38 20.49 266.27 293.28 307.02 321.78 347.77 1.00 440  
##   
## 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 = var(deviance)/2)  
## pD = 209.7 and DIC = 517.1  
## DIC is an estimate of expected predictive error (lower deviance is better).

Look at the model output above for the estimate of psi. Bayesian analysis does not produce a single paramter estimate with an error (as in frequentist analysis), but rather a posterior distirbution for the parameter. So, “mu.vect” is the mean of that distribution and “sd.vect” is the standard deviation. The columns from “2.5%” through “97.5%” are the quantiles of the distribution and “n.eff” is the effective sample size (i.e. the number of iterations corrected for autocorrelation, but we won’t go into that now). Finally, “Rhat” is a measure of how well the mcmc chains have converged and is a standard model diagnostic to make sure it has run enough iterations to get a good estimate. An Rhat < 1.1 is considered good, so it looks like the chains have converged. The mean parameter estimates are and `, which is close to our simulated values of 0.7 and 0.4, respectively.

### A More Realistic Occuapncy Model

Next we will simulate some data where occupancy and detection probability are both dependent on covariates, again following Kéry (2010). The simulated occupancy relationship will be a quadratic response to a vegetation index ranging from -1 to 1, and the detection probability will have a negative response to the same index. Therefore, the model of the occupancy relationship is:

and

The model for detection probability is:

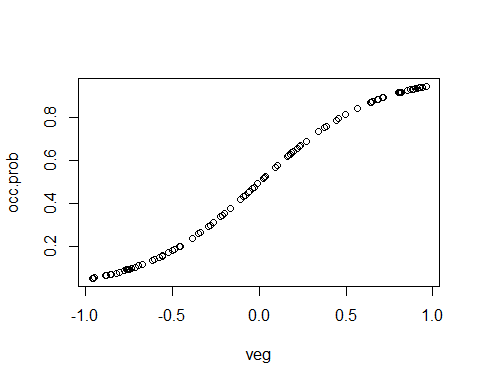
and

Now we will simulate the data.

veg <- sort(runif(n = R, min = -1, max = 1))  
  
alpha.occ <- 0  
beta1.occ <- 3

The “true” occupancy is generated by applying the parameters to the veg data at each site. The result of the equation is tranformed to a probability using the inverse-logit equation where .

occ.prob <- exp(alpha.occ + beta1.occ\*veg)/(1 + exp(alpha.occ + beta1.occ\*veg))  
true.occ <- rbinom(R, 1, occ.prob)  
plot(veg, occ.prob)

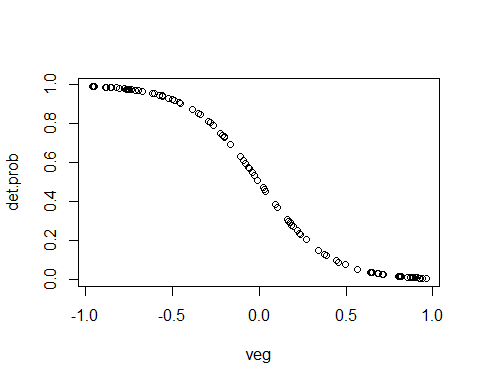


sum(true.occ)/100 # The true occupancy rate

## [1] 0.45

The simulated detection probability follows the same format.

alpha.p <- 0  
beta1.p <- -5  
  
det.prob <- exp(alpha.p + beta1.p\*veg)/(1 + exp(alpha.p + beta1.p\*veg))  
y <- matrix(NA, R, T)  
for(i in 1:R){  
 y[i, ] <- rbinom(T, 1, det.prob[i]\*true.occ[i])  
}  
plot(veg, det.prob)



So, while occupancy probability increases with the vegetation index (to a point), detection probability decreases. The BUGS model is simply an extension of the previous model, with prior distirbutions on the paramters in the regression equation rather than on and .

sink("model\_2.txt")  
cat("  
model{  
 # The prior disturbutions on psi and p  
 alpha.occ ~ dnorm(0, 0.0001)  
 beta1.occ ~ dnorm(0, 0.0001)  
 beta2.occ ~ dnorm(0, 0.0001)  
   
 alpha.p ~ dnorm(0, 0.0001)  
 beta1.p ~ dnorm(0, 0.0001)  
   
 # The likelihood  
 for(i in 1:R){  
 # Model 1: the occupancy process  
 logit(psi[i]) <- alpha.occ + beta1.occ\*veg[i]  
 z[i] ~ dbern(psi[i])  
   
 # Model 2: the detection process, conditional on occupancy  
 for(j in 1:T){  
 logit(p[i, j]) <- alpha.p + beta1.p\*veg[i]   
 y[i, j] ~ dbern(z[i]\*p[i, j])  
 }  
 }  
 # Derived quantities  
 occ.fs <- sum(z[])  
}  
", fill = TRUE)  
sink()

A couple of things to notice about the model above:

1. The priors on the parameters follow a very wide normal distirbution with mean 0 and = 10,000. The reason the variance parameter in the model priors is 0.0001 is that BUGS uses a parameter to describe variance, where . Don’t ask me why, it’s just what they do.
2. It includes a derived paramter, “occ.fs”, which is an estimate of the total number of occupied sites (out of 100). Thus, the model estimates this number at each iteration and part of the ouput is a posterior distribution of the estimated total occupancy (the sum of all ).

# Compile the data  
data <- list(R = R, T = T, y = y, veg = veg)  
  
# Set the parameters to monitor  
params <- c("alpha.occ", "beta1.occ", "alpha.p", "beta1.p", "occ.fs")  
  
# Specify the initial values - this is critical in JAGS, and is why we use the naive occupancy as initial values for z  
zinit <- apply(y, 1, max)  
inits <- function(){list(z = zinit, alpha.occ = rnorm(1, 0, 1), beta1.occ = rnorm(1, 0, 1), alpha.p = rnorm(1, 0, 1), beta1.p = rnorm(1, 0, 1))}  
  
nc <- 3   
ni <- 5000   
nb <- 500   
nt <- 10

out2 <- jags(data, inits, params, model.file = "model\_2.txt", n.chains = nc, n.iter = ni, n.burnin = nb, n.thin = nt)  
print(out2, digits = 2)

## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 300  
## Unobserved stochastic nodes: 105  
## Total graph size: 1219  
##   
## Initializing model

## Inference for Bugs model at "C:/Users/crosby/AppData/Local/Temp/RtmpMfoT1f/model1cdc192a3a99.txt", fit using jags,  
## 3 chains, each with 5000 iterations (first 500 discarded), n.thin = 10  
## n.sims = 1350 iterations saved  
## mu.vect sd.vect 2.5% 25% 50% 75% 97.5% Rhat n.eff  
## alpha.occ -0.07 0.47 -0.92 -0.40 -0.10 0.22 0.95 1.00 900  
## alpha.p -0.41 0.37 -1.14 -0.65 -0.42 -0.18 0.30 1.00 1400  
## beta1.occ 3.21 1.08 1.36 2.44 3.12 3.92 5.44 1.01 220  
## beta1.p -3.89 0.76 -5.50 -4.34 -3.85 -3.39 -2.54 1.00 1400  
## occ.fs 47.48 5.83 35.00 44.00 48.00 51.75 58.00 1.00 1300  
## deviance 104.31 9.05 89.19 97.61 103.57 110.14 124.16 1.00 1400  
##   
## 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 = var(deviance)/2)  
## pD = 41.0 and DIC = 145.3  
## DIC is an estimate of expected predictive error (lower deviance is better).

### Resources

I hope this tutorial has been helpful. For more in-depth reading on fitting hierarchical models using Bayesian methods see Kéry (2010), Kéry and Royle (2016), and Royle and Dorazio (2008).

**Literature Cited**

Kéry, M. 2010. Introduction to WinBUGS for Ecologists: a Bayesian Approach to Regression, ANOVA, Mixed Models and Related Analyses. Academic Press, New York, New York, USA.

Kéry, M., and J. A. Royle. 2016. Applied Hierarchical Modeling in Ecology: Analysis of Distribution, Abundance, and Species Richness in R and BUGS. Volume 1, Prelude and Static Models. Academic Press, San Diego, CA, USA.

Royle, J. A., and R. M. Dorazio. 2008. Hierarchical modelling and inference in ecology. Academic Press, San Diego, CA, USA.