An Occupancy Model Where ψ and p Vary Among Sites

EEB 5894-003 "Detection, Occurrence, and Abundance"
Lab: Tuesday, September 29, 2015

This week we will expand our Bayesian analysis to add covariates on ψ and p. In other words, we will change the model such that ψ and p can vary among sites and we will use covariates to try to explain differences in ψ and p among sites.

We are interested in the following covariates for ψ:

- Elevation of each survey site
- Burn severity at each survey site
- Pre-fire canopy cover at each survey site

We are also interested in the effect of Julian date on our ability to detect individuals when they are present at a site.

Each of these covariates was explained in class.

To get started, download the "yrwa_detection.csv", "site_covariates.csv", and "week5_script_skeleton.R" files from HuskyCT. Remove the site names from the YRWA detection data.

1. Editing the JAGS Model File

We will need to make several modifications to the JAGS model file from last week to incorporate covariates. First let's modify the equations describing the likelihood of both ψ and p. The likelihood equations for a model including the covariates described above are:

$$z_j \sim Bernoulli(\psi_j) \qquad \textit{state process}$$

$$logit(\psi_j) = \beta_0 + \beta_1 Elevation_j + \beta_2 Burn\ Severity_j + \beta_3 Canopy\ Cover_j$$

$$y_{j,k} \sim Bernoulli(z_j p_j) \qquad \textit{observation process}$$

$$logit(p_j) = \alpha_0 + \alpha_1 Julian\ Date_j$$

There are two important differences between these equations and the equations from last week. First, both ψ and p have subscripts j because both ψ and p can differ among sites. Second, we now use the logit link to make ψ and p functions of the covariates.

1.1. *Add the code for the likelihood of the state (i.e., the occupancy) process. Note that the logit part of the equation is deterministic so it does not use the tilde (~).*

1.2. *Add the code for the observation process. Note that as with last week, you cannot do the calculation $z_{j*}p_j$ in the dbern() function. Also, it is easiest to just index p by both j and k, even though Julian date is the same among the three replicate surveys at a site. So, your likelihood will look more like the following (all of which can go in the k loop that loops through the repeat surveys):*

$$y_{j,k} \sim Bernoulli(\mu_{j,k})$$

$$\mu_j = z_j p_{j,k}$$

$$logit(p_{j,k}) = \alpha_0 + \alpha_1 Julian\ Date_j$$

Now we can define the priors. We have added a lot more variables this week; hence, we will also need a lot more priors. We need priors for the intercepts and coefficients in the linear model for ψ and p. All of these values can range between negative infinity and infinity and we don't have any prior information to inform us on which values are more likely. Hence, we can describe our priors with a flat normal distribution. In JAGS this can be done using the code *dnorm(0, 0.001)*. This specifies a normal distribution centered on zero with a very large variance (remember JAGS uses the precision rather than the standard deviation to describe the variance, where the precision is the inverse of the standard deviation…that's why 0.001 describes a distribution with a very large variance).

1.3. *Use the dnorm() function to provide the priors for the intercept and three coefficients in the occupancy likelihood.*

1.4. *Use the dnorm() function to provide the priors for the intercept and coefficient in the observation likelihood.*

2. RUNNING THE JAGS MODEL FROM R

Now that we have our model specified, we just need to prepare our data and few other parameters to run the model from R. First, load the packages *rjags* and *R2jags*.

Now, let's scale the covariates. Scaling covariates serves two purposes. First, scaling the covariates can decrease the number of iterations you need to run the MCMC algorithm to make the chains converge, which saves you time running the model. Second, it changes the interpretation of the intercept in the model for both ψ and p. For example, without scaling the covariates, the intercept for ψ is interpreted as the occupancy probability at a site where all the covariates are equal to zero. However, when the covariates are scaled the intercept is interpreted as the occupancy at a site when all the covariates are at their mean value.

Covariates are often scaled as:

$$x_{scaled} = \frac{x - \bar{x}}{SD(x)}$$

Of course, R has a function for that called *scale()*.

2.1. *Save a scaled version of each of the four covariates using the scale function as follows: new.var = scale(data$covar), where new.var is the name of your new R object, data is the covariate data you imported, and covar is the name of the covariate you want to scale.*

Now let's put all our data in a list that we will pass to JAGS. Remember, you specify a list in R using the code *list(name1 = x1, name2 = x2, name3 = x3)*, where name is the name of the variable used in the JAGS model text file (these names must match), and x is the object from R or a constant (e.g., T = 3). This list needs to include the YRWA detection data, the number of sites, the number of replicate surveys, and each of your scaled covariates. Each covariate is provided separately in the list.

2.2. *Create a list in R containing the YRWA detection data, the number of survey sites (N), the number of replicate surveys at a site (T), and each of the 4 scaled covariates. Refer to your JAGS model file code to make sure you provide the right names for each of these covariates.*

Now, provide the initial values. This can be done using the same function we created last week.

```
inits = function(){list(z = apply(data, 1, max))}
```

2.3. *Add the code to provide the initial values for whether the site is occupied.*

Last, create a list of variable names that you want to monitor. I suggest monitoring the intercept and coefficients for both ψ and *p*.

Now you can pass all of this information to JAGS using the function *jags()* as follows:

```
jags(data = your list of data,
      model.file = “the name of your model file”,
     inits = the inits function you created,
     parameters.to.save = the vector of parameter names you created,
     n.burnin = the number of initial MCMC steps to discard,
     n.iter = the number of MCMC steps to run after the burn in,
     n.thin = the thinning rate (a positive integer),
     n.chains = the number of chains)
```

2.4. *Use the jags() function to run your model using the following MCMC settings: n.burnin = 1000, n.iter = 3000, n.thin = 2, n.chains = 3. Save your results as an R object.*

3. Interpreting and Visualizing the Model Results

As always with Bayesian analysis, the first thing we want to do is see if the chains converged for each parameter. You can visualize each chain using the traceplot() function in R.

3.1. *Use the traceplot(fitted_jags_object) function on your fitted model object from step 2.4 to visualize the chains for each parameter. In the plots, each chain is shown in a different color. If the chains have converged, the colors should look well mixed. If one of the chains appears to result in a different estimate for any of the parameters, then the chains have not converged. If the chains have not converged you will need to run more MCMC iterations and increase the burn-in size.*

Now we can check to see if the relationship between the covariates and ψ and p are significant (i.e., the coefficients are greater than 0). We can do this by looking at the 95% credible intervals (i.e., Bayesian confidence intervals) for each of the coefficients. This interval is provided as the 2.5% and 97.5% quantiles in the summary of the JAGS model fit object you created in step 2.4. If the 95% credible interval does not contain zero, then we can say we are 95% confident that the true parameter estimate is not zero given the data.

3.2. Look at your model fit object (by typing the name in the R console and hitting enter). Which covariates are significant and which are not?

Last, let's interpret a few estimates from the model. First, let's look at the mean and 95% credible interval of the occupancy and detection probability at an average site (i.e., a site with the average value for all covariates). Because, we scaled the covariates this is just the intercept back transformed to the probability scale. You can use the *plogis()* function to transform the intercept estimates from the model to the probability scale. You can extract the 2.5%, 50%, and 97.5% quantiles from the posterior of a coefficient using the following code: *quantile(fit$BUGSoutput$sims.list$coef, c(0.025, 0.5, 0.975)),* where *fit* is the name of your fitted JAGS model object and *coef* is the name of the coefficient you want to extract from the JAGS model output.

3.3. Use the code above to look at the median and 95% credible interval for occupancy and detection at an average site.

We can also plot the relationships for the significant variables. The code for this is mostly provided for you. You just need to save the posteriors from the JAGS model to R objects for use in the code at the end of the script.

3.4. Add the code to extract the posterior for the specified coefficients. You can extract covariates using the code fit$BUGSoutput$sims.list$coef, where fit is the name of your fitted JAGS model object and coef is the name of the coefficient you want to extract from the JAGS model output. Then run the code in the rest of the script to make the plots. How do elevation and burn severity affect occupancy? How does Julian date affect detection probability?

4. BONUS

4.1. Add lines for the 95% credible intervals on the plots below. You can use the function lines() to add lines to an existing plot.

4.2. Run this covariate model in unmarked.