Dynamic Occupancy Models

Colin Lewis-Beck

July 16, 2017

Fitting Dynamic (MultiSeason) Site Occupancy Models for one species is more challenging given the additional dimension of time (or season), where seasons are linked via a Markovian transition. Below is an outline of the type of models I hope the nimble.dynam.occ function will fit. In addition to allowing random effects, using the new MCMC algorithms for these models developed by the NIMBLE team could speed up estimation. First is an outline of the most generic dynamic site occupancy model.

Let $y(i, j, t)$ by the detection/non detection variable for site i, replicate survey j, and season t. Let $z(i, t)$ be the latent occurrence state for site i over season t. The model has two stages:

$$z_{i,t} \sim Bern(\psi_{i,t})$$

$$y_{i,j,t} \sim Bern(z_{i,t} p_{i,j,t})$$

The initial ecological state at, $t = 1$ is $z_{i,1} \sim Bern(\psi_{i,1})$

After the first season, we have the following Markov structure where ϕ is the site survival probability and γ is the site colonization probability.

$$z_{i,t} \sim Bern(z_{i,t-1}\phi_{i,t-1} + (1 - z_{i,t-1})\gamma_{i,t-1}))$$

$$y_{i,j,t} \sim Bern(z_{i,j} p_{i,j,t})$$

Each of the following parameters can be specified via a linear model: ψ, ϕ, γ, p

1) $logit(\psi_{i,1}) = \alpha + \beta x_i$
2) $logit(\phi_{i,t}) = \alpha + \beta x_{i,t}$
3) $logit(\gamma_{i,t}) = \alpha + \beta x_{i,t}$
4) $logit(p_{i,j,t}) = \alpha + \beta x_{i,j,t}$

I imagine the function nimble.dynam.occ will take model arguments for the 4 models listed above, as well as take data for y and data for the site covariates (i), year/site covariates (i, t) and observation covariates (i, j, t). This input structure is similar to the unmarked package function colext.

A challenge will be to define the code generation in a way that will allow us to use NIMBLE's new MCMC algorithms for these latent state space models, as well as work with lmPred and nim_glm, which are generic functions that expand model statements, add priors, etc. A general type of code structure for this type of model is:

```
for (i in 1:nsite){
  logit(psi[i]) <- alpha + beta*x[i]    #Model for Season 1 latent state
  z[i, 1] <- psi[i]
  for (t in 2:nseason){
    muZ[i,t] <- z[i, t-1]*phi[i, t-1] + (1-z[i, t-1])*gamma[i,t-1]
    z[i, t] ~ dbern(muZ[i,t])
  }
}
```

```
#Model for Survival and Colonizing probabilities
for (i in 1:nsite){
  for (t in 1:nseason){
    logit(gamma[i,t]) <- alpha + b*x[i,t]
    logit(phi[i,t]) <- alpha + b*x[i,t]
    }
}

for (i in 1:nsite){
  for(j in 1:nrep){
    for(t in 1:nseason){
      logit(p(i,j,t)) <- alpha + b*x[i,j,t]  #Model for Detection Probability
      muy[i,j,t] <- z[i,t]*p[i,j,t]
      y[i,j,t] <- dbern(muy[i,j,t])
    }
  }
}

#Note: Priors would also be generated for all Model Parameters
```

One thing I'm uncertain is how to maintain compatability with the TidyR (long) data format and the double indexing with respect to site and time that is required by these models. For example, we could use the nim_glm expansion one of the models for gamma.

```
gamma[1:site*season] ~ nim_glm(1 + season, factors = "season", link = logit, priors = priors)
```

However, in the latent state space formula we need to be able to split and lag the gamma vector as gamma[i, t-1], e.g.

```
muZ[site,t] <- z[site, t-1]*phi[i, t-1] + (1-z[site, t-1])*gamma[i,t-1]
```

It might be possible to reconstruct the matrices for the parameters in a way that is compatible with the double indexing after running the nim_glm statements. Also, consideration for the format of the different MCMC algorithms, specifically, the filtering algorithm, might require different formatting of the latent states and parameters. It would be nice if we could offer users a couple different MCMC algorithms as options in the function call, and then automatically reformat the model code appropriately.