Phylogenetic occupancy models integrate imperfect detection and phylogenetic signal to analyze community structure

Luke O. Frishkoff, Perry de Valpine, Leithen K. MGonigle

September 26, 2016

1 Introduction

The following text provides instructions on how to use the Phylogenetic Occupancy Model (POM), as presented in Frishkoff et al. (2016). The POM enables users to incorporate phylogenetic information directly into occupancy models. In this framework, phylogenetic information can be incorporated into species' individual rates of occupancy and/or into their responses to an environmental gradient. We provide (and summarize below) scripts that generate a sample data-set on which the POM can be run. Data simulation is conducted in R (R Core Team, 2016) and occupancy models are implemented in JAGS (Plummer et al., 2003) via R using the packages `rjags` and `R2jags`.

2 Phylogenetic Occupancy Model (POM)

Prior to running the POM, you will need to install JAGS (http://mcmc-jags.sourceforge.net/) as well as install and load the following libraries:

```
library(ape)
library(MASS)
library(rjags)
library(R2jags)
```

The data simulation code is available as a function in the script `simulate_data.R` and the JAGS model code for the POM, again written as a function, is contained in `pom.R`.
The first of these scripts contains a function `make.data.JAGS` that generates the necessary components required for the JAGS model. This function takes a number of optional arguments (summarized in Table 1) and returns a list with 11 elements (summarized in Table 2) that are formatted as required by the POM.
We also provide a script, `main.R` , that loads the above scripts, simulates a data-set, and runs the POM. We will summarize the important parts of that script here. First we set the seed using `set.seed(12345)` for reproducibility.

To simulate the data, one must first construct (or input) a phylogenetic tree. Here, we construct a random coalescent tree with 32 species:

```
num.sp <- 32
tree.b <- rcoal(num.sp, tip.label=1:num.sp)
```

This tree can then be passed in to `make.data.JAGS` , along with any other arguments (as specified in Table 1). For example:

```
dd.model <- make.data.JAGS(nsp=num.sp,
                           tree=tree.b,
                           beta.trait=1,
                           lambda=1,
                           sigma.psi.beta=1,
                           sigma.psi.site=1,
                           sigma.psi.yr=1,
                           sigma.p.0=1,
                           mu.p.0=-1)
```

As is standard for a JAGS model, one must also construct initialization values for the MCMC (inits). Here, we specify inits only for the latent true occupancy state, `Z` . Doing so ensures that there are no nonsensical values of Z upon initialization (e.g. `Z = 0` when a species was observed and `X = 1`).

```
  z.init <- dd.model$Z # This does need a $ right?
  my.inits <- function() {
    list(Z=z.init)
  }
```

Note that dd.model$Z is generated automatically within the `make.data.JAGS` function.
We then construct the data object required for JAGS which includes a list of model parameters that we would like to track:

```
  dd <- list(data=dd.model,
             inits=my.inits,
             params=c(’p.0’,
                      ’p.beta’,
                      ’mu.p.0’,
                      ’mu.p.beta’,
                      ’sigma.p.0’,
                      ’sigma.p.beta’,
                      ’psi.0’,
                      ’psi.beta.sp’,
                      ’mu.psi.beta’,
                      ’sigma.psi.beta’,
                      ’sigma.psi.site’,
                      ’sigma.psi.year’,
                      ’beta.trait’,
                      ’lambda’))
```

Finally, we can run the POM using the function `run.R2jags.model` which is specified in the `pom.R` script.

```
 res.lam <- run.R2jags.model(dd, ni=1000, nb=100, nt=2, nc=3)
```

Here, `ni` specifies the number of iterations, `nb` the number of burn in iterations, `nt` the thinning rate, and `nc` the number of chains. Once the model has finished running one could, for example, inspect the posteriors for all tracked parameters:

```
 sum.lam <- res.lam$BUGSoutput$summary
 cols <- c(’mean’, ’2.5%’, ’97.5%’, ’Rhat’, ’n.eff’)
 round(sum.lam[,cols], 3)
```

One could also investigate the model-estimated lambda and compute a bounded region of highest posterior probability:

```
 lamdba <- sum.lam[’lambda’,]
 sm <- res.lam$BUGSoutput$sims.matrix
 source(’bounded_hpp.R’)
 bounded.hpp(sm[,’lambda’])
```

In this case, the data we simulated had a `lambda=1.0` and we find a mode of 1.0 with a region of highest posterior probability density ranging from $(0.601, 1.000)$.
One may wonder if the detection component of the model is appropriately specified, or if it is overly complex. Perhaps individual species do not vary in how their detectability changes along the environmental gradient? One can examine the `sigma.p.beta` term to address this question.

```
bounded.hpp(sm[,’sigma.p.beta’])
```

Because the HPD intersects with 0, there is not strong evidence for this term being included in the model. If the user wishes to proceed with model selection in order to best specify the terms in the POM then one could alter the code in the `pom.R` file to eliminate this random effect. For example, line 23-24:

```
  sigma.p.beta ~ dunif(0, 20)
  tau.p.beta <- 1/(sigma.p.beta* sigma.p.beta)
```

could be deleted, and line 29 replaced with a simple fixed effect:

```
   p.beta[sp] <- mu.p.beta
```

Then the POM could be rerun as above, and the `mu.p.beta` parameter queried to assess whether this term could in turn be dropped from the model.

References

Frishkoff, L. O., P. de Valpine, and L. K. M'Gonigle, 2016. Phylogenetic occupancy models integrate imperfect detection and phylogenetic signal to understand the drivers of community structure. Ecology XX.

Plummer, M. et al., 2003. Jags: A program for analysis of bayesian graphical models using gibbs sampling. *in* Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003). March, Pp. 20–22.

R Core Team, 2016. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Name	Description	Default value
`nsp`	Number of species.	64
`nsite`	Number of sites.	15
`nyr`	Number of years.	5
`nrep`	Number of samples within a year.	5
`mu.psi.0`	Mean intercept for occupancy.	0
`sigma.psi.0`	Variation in intercepts for occupancy.	0.5
`mu.p.0`	Mean interecpt for detectability.	0
`sigma.p.0`	Variation in intercepts for detectability.	0.5
`mu.psi.beta`	Mean response to environmental gradient.	0
`sigma.psi.site`	Variation in site random effect for occupancy.	0.1
`sigma.psi.yr`	Variation in year random effect for occupancy.	0.1
`psi.0.trait`	Effect of trait on mean occupancy (i.e., slope of occupancy with regard to species' traits).	0
`beta.trait`	Effect of trait on species' responses to the environment.	0
`tree`	Phylogentic tree.	NULL
`sigma.psi.beta`	Variation in species' responses to the environment. The width of the covariance matrix that is used to draw species' responses to the environment is scaled by this value.	1
`lambda`	Weighting on phylogenetic covariance matrix (e.g., with `lambda=1` , the covariance matrix is given full weight and with `lambda=0` , the model reduces to a standard random effect).	1
`lambda.trait`	Degree to which the trait is phylogenetically conserved.	0
`mu.p.beta`	Mean detection response to environment.	0
`sigma.p.beta`	Variance of detection responses to environment.	0
`psi.image`	Plot mean occupancy probability while creating data.	FALSE
`occ.image`	Plot mean occupancy while creating data.	FALSE
`det.image`	Plot mean detection probability while creating data.	FALSE
`plot.tree`	Plot tree colored by species' occupancy slopes while creating data.	FALSE

Table 1: Description of optional arguments used to generate data in the function `make.data.JAGS` .

Name	Description	Format
`inputs`	Arguments used to generate data.	List
`nsp`	Number of species.	Integer
`nsite`	Number of sites.	Integer
`nyr`	Number of years.	Integer
`nrep`	Number of samples for each species at each site in each year.	Array (`nsp` × `nsite` × `nyr`)
`trait`	Species' trait values.	Vector (`nsp`)
`env`	Value along the environmental gradient for each site.	Vector (`nsite`)
`X`	Species' observation records.	Array (`nsp` × `nsite` × `nyr` × `nrep`)
`Z`	Species' true occupancy.	Array (`nsp` × `nsite` × `nyr`)
`ID`	Identity matrix.	Array (`nsp` × `nsp`)
`VCOV`	Phylogenetic Variance-Covariance matrix.	Array (`nsp` × `nsp`)

Table 2: Description of data structures created by `make.data.JAGS` .