A users' guide to apply the presence-only joint distribution framework for single species

Juan Escamilla Molgora, Luigi Sedda, Peter Diggle, Peter Atkinson

November 11, 2021

Abstract

This document is a guide to use the models described in the paper: *A joint distribution framework to improve presence-only species distribution models by exploiting opportunistic surveys* ([1]) for modelling species distributions with presence-only data.

The presented framework proposes three bayesian models for inferring single species distributions using solely observations of species occurrences (i.e. presence-only data). The fundamental idea of the framework is that the observed occurrences (Y) are determined not only by the environmental conditions (niche) suitable for the species to thrive but also accounts for the bias in which these observations were collected (i.e. sampling effort). As such, the occurrences are specified as realisations of a join process composed of environmental conditions (P_Y) and sampling effort (P_X). These two components are explicitly defined with a model-based approach ([2]) to specify two types of covariates, one for P_X and other for P_Y. Both processes are specified as mixed models. That is, they are composed of a fixed effect (i.e. $E(P_X) = g(d_X; \beta_X)$ and $E(P_Y) = g(d_Y; \beta_Y)$) and a random effect that models the variation between observations across space. For the purposes of this example (and the presented results in the manuscript) we consider $g(d_X; \beta_X)$ to be linear, that is: $g(d_X; \beta_X) = \beta_X^t d_X$, where d_X is the vector of covariates associated to a given observation. For a clear description of the model refer to the manuscript and, specifically, to the supplementary materials.

The dependencies between processes P_X and P_Y are specified in terms of the covariance between all observations. That is, by their random effects; R_X and R_Y respectively. We proposed three forms to model these dependencies. Model I assumes R_X and R_Y are independent. Model II assumes that $R_X = R_Y$ and Model III assumes both random effects are correlated. The three

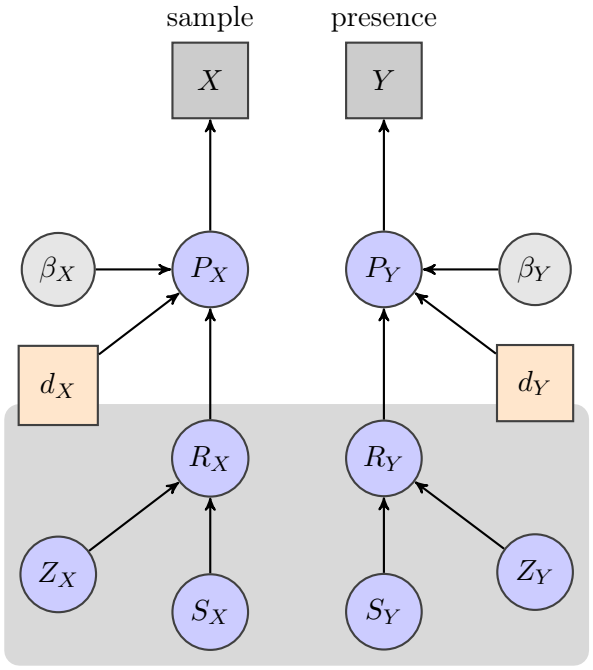

Figure 1: DAG for model I

models specify a conditional autoregressive (CAR) ([3]) model for defining the spatial random effect. Therefore, this framework is only suited for data aggregated in areas tilled over a region (i.e. spatial lattices).

In the following sections we will apply models I, II and III in the **R** programming language using the package CARBayes ([4]). The data used was obtained from GBIF using the dataset ([5]). This document is intended to serve as a guide for modelling species distributions using the aforementioned framework. For further reference of the Bayesian methodologies, diagnoses and deeper analysis of lattice-based models refer to ([4, 6])

1 Model I

This model assumes that the ecological suitability distribution (Y) and sampling effort X process are independent. The structure of the model is visualised as a directed acyclic graph (DAG) in the figure 1. For a detail explanation of the model refer to the supplementary materials of the manuscript.

1.1 Running the model

We begin by defining the formulas for the sampling effort ($\text{formula}_{\text{sample}}$) and the environmental niche ($\text{formula}_{\text{presence}}$). As we are dealing with only presence-absence realisations, we define a bernoulli process. That is, a binomial distribution with $k = 1$. This can be specified with a 'trials' vector built on only ones.

```
formula_sample=sample~Disttoroadm+Populationm
formula_presence=species~Elevationm+MeanTempm
n <- nrow(TDF)
trials <- rep(1,n)
```

We define the running settings for sampling the posterior distribution. In this case we specify 1,000 iterations after burnin (i.e. initialisation to reach stationarity of the Markov chain) of 70,000. To reduce hoarding data in memory we choose a thinning of 10. The postburnin parameter defines the number of iterations after burnin for each process but before saving the final sample of 1,000 iterations. Postburnin defines and extra burn-in iterations.

```
n.sample = 100000
burnin=70000
thin = 10
postburnin = burnin + 10000

model1  <- joint.binomial.bymCARModel1(formula_S = formula_sample,
                                       formula_P = formula_presence,
                                       n.sample=n.sample,
                                       data = DataFrame,
                                       burnin=burnin,
                                       postburnin=postburnin,
                                       thin=thin,
                                       verbose=TRUE)
```

```
TRUE
```

In model I, the processes S (sample) and P (environment) are independent. The object model1 contains these independent components stored as attributes S and P respectively. To inspect the posterior summary statistics we can use the attribute summary.results on each process.

```
model1$P$summary.results
```

	Median	2.5%	97.5%	n.sample	% accept	n.effective	Geweke.diag
(Intercept)	-14.6010	-22.0458	-5.6759	10000	46.6	66.8	1.4
Elevationm	0.0016	0.0000	0.0031	10000	46.6	83.1	0.5
MeanTempm	-0.0428	-0.3228	0.2136	10000	46.6	104.9	0.6
tau2	6.9339	3.8265	15.0163	10000	100.0	45.5	-1.1
sigma2	0.0312	0.0029	0.3400	10000	100.0	8.1	1.4

1.2 Modelfit statistics

Model fitting criteria can be inspected with the argument (modelfit). Available statistics are: Deviance information criteria (DIC), Watanabe-Akaike information Criterion (WAIC), log marginal predictive likelihood (LMPL), estimated effective number of parameters (p.d.) and loglikelihood.

```
model1$summary.results #For general modelfit of both models simultaneously.
model1$P$modelfit
model1$S$modelfit
```

1.3 Convergence of the Markov chain by visual inspection

```
library(coda)
params.samples <- mcmc.list(model1$P$samples$beta)
plot(params.samples)
```

1.4 Cross-validation

In this subsection we will perform a cross-validation of model I. Cross validation of CAR models is not straight forward due to its spatial structure (lattice). The approach for this would be with data augmentation and the k-fold method ([7]). In this case we will randomly (without replacement) select 1/10 of the observations (sites) as missing data, fit the model, estimate the probability of Y and X on these sites and repeat this operation 10 times until we obtain a whole set of estimated observations. Afterwards we will calculate the the Receiver Operator Characteristic and its Area Under the Curve (i.e. AUC-ROC, also called ROC-curve).

We will load the script $\text{init}_{\text{data.R}}$

1.5 Setting the working directory.

Change accordingly to your installation.

```
setwd('/main/app/external_plugins/biospytial_rwrapper/CAR-1SDM/')
source('R/preprocess_data.R')
# load Model 1
source("joint.binomial.bymCARModel1.R")
```

A brief data exploration shows the number of relative absences and presences ($\text{npresence}_{0,\text{npresece1}}$) in the "presences" process Y, the in the sampling process X (i.e. nsample_0 and nsample_1, respectively). Missed data (no observations) are shown as $\text{n}_{\text{misspresence}}$ and $\text{n}_{\text{misssample}}$. for Y and X respectively.

```
ntot = length(DataFrame$sample)
npresence_1 = length(na.omit(DataFrame$species[DataFrame$species == 1])) / ntot*100
nsample_1 = length(na.omit(DataFrame$sample[DataFrame$sample == 1]))/ntot*100
npresence_0 = length(na.omit(DataFrame$species[DataFrame$species == 0]))/ntot*100
nsample_0 = length(na.omit(DataFrame$sample[DataFrame$sample == 0]))/ntot*100
n_miss_presence = sum(is.na(DataFrame$species))/ntot*100
n_miss_sample = sum(is.na(DataFrame$sample))/ntot*100
cbind(npresence_0,npresence_1,nsample_0,nsample_1,n_miss_presence,n_miss_sample)
```

```
     npresence_0 npresence_1 nsample_0 nsample_1 n_miss_presence n_miss_sample
[1,]    63.02956    8.399015  10.59113  71.42857        28.57143       17.9803
```

1.6 Preparing the cross-validation

1. We begin by loading necessary packages:

   ```
   library(pROC)
   library('caret')
   ```

   ```
   Error in library(pROC) : there is no package called ‘pROC’
   Error in library("caret") : there is no package called ‘caret’
   ```

2. Create partitions for cross validation using the K-fold method (k=7) and prepare the dataframe to receive results.

   ```
   nonas = which(! is.na(DataFrame$species) )
   Y_withoutNA = DataFrame$species[nonas]
   ```

```
validate = createFolds(y = Y_withoutNA, k=7, returnTrain = FALSE)
DataFrame$presences <- DataFrame$species
l <- list()
i = 1
DataFrame$predicted_values_CV <- NA
DataFrame$predicted_valuesBernoulli <- NA
```

```
Error in which(!is.na(DataFrame$species)) : object 'DataFrame' not found
Error: object 'DataFrame' not found
Error in createFolds(y = Y_withoutNA, k = 7, returnTrain = FALSE) :
  could not find function "createFolds"
Error: object 'DataFrame' not found
Error in DataFrame$predicted_values_CV <- NA :
  object 'DataFrame' not found
Error in DataFrame$predicted_valuesBernoulli <- NA :
  object 'DataFrame' not found
```

3. Define the formulas for the presence (environmental) process Y and the sampling effort X

```
formula_sample =  sample ~ Disttoroadm + Populationm
formula_presence = species ~ Elevationm + Precipitationm
```

4. Cross validation

   - First we define the parameters for sampling the posterior distribution (MCMC).

```
n.sample = 20000
burnin=15000
postburnin = burnin + 1000
```

   - Perform the cross-validation

```
for (fold in validate) {
  observed.presences <- DataFrame$species[fold]
  ## Substitue by NA
  DataFrame$species[fold] <- NA
  results  <- joint.binomial.bymCARModel1(formula_S = formula_sample,
                            formula_P = formula_presence,
```

```
                                        n.sample=n.sample,
                                        data = DataFrame,
                                        burnin=burnin,
                                        postburnin=postburnin,
                                        thin=thin,
                                        verbose=TRUE)

            DataFrame$species <- DataFrame$presences
            ## Return original values
            DataFrame$predicted_values_CV[fold] <- results$fitted.values[fold]
            predicted.probability = results$fitted.values[fold]

         ## Generate Bernoulli sample [ Only for the fold data set]
         print("Generating bernoulli sampling...")
         post.joint = data.frame(results$samples$fitted.joint[fold])
         ptot <- post.joint %>% mutate_all(function(p) rbernoulli(1,p))
         sumpt <- colSums(ptot)
         nsamples = dim(ptot)[1]
         ProbPS <- sumpt / nsamples

         DataFrame$predicted_valuesBernoulli[fold] <- ProbPS
         i = i + 1
     }

Error in validate : object 'validate' not found
```

1.6.1 Plot the RO curve

```
pROC_obj <- roc(DataFrame$presences,DataFrame$predicted_values_CV,
                smoothed= TRUE,
                ci = TRUE, ci.alpha=0.9, stratified=FALSE,
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(pROC_obj)
plot(sens.ci,type="shape",col="lightblue")
plot(sens.ci, type="bars")
```

```
org_babel_R_eoe
```

Procedure for cross-validating models II and III can be performed similarly.

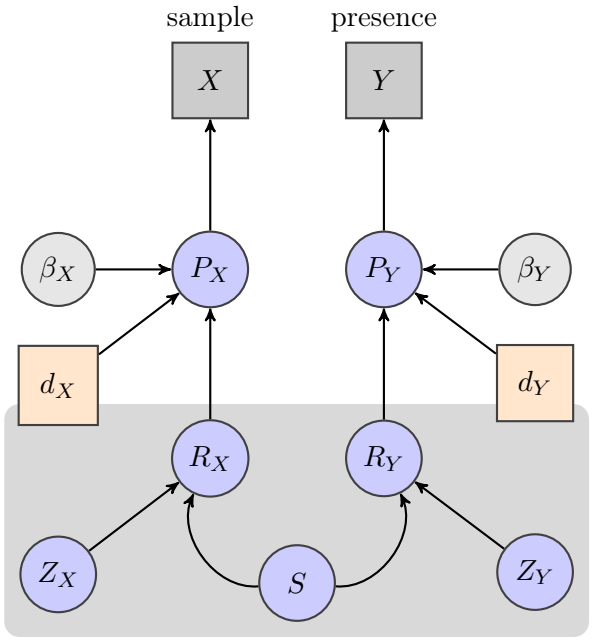

Figure 2: DAG for model II

2 Model II

This model assumes that the ecological suitability distribution (P) and sampling effort (S) processes are independent when conditioned to a common spatial structure (G) (a Gaussian Markov random field). The structure of the model is visualised as a directed acyclic graph (DAG) in the figure 2. For a detail explanation of the model refer to the supplementary materials of the manuscript.

2.1 Running the model

To run the model we first need to transform the data, specifically, the covariates matrix (i.e. design matrix). The transformation allows the fitting of specific covariates for each process (i.e. an environmental set of covariates and another set for the sampling effort).

To do this, we stack the design matrices of S and P and extend the number of columns to the sum of both processes. That is, the number of columns of the resulting matrix is the sum of the columns of P and S (not counting the response variables). We begin by selecting the covariates for

each process following a standard formula syntax (one formula per process).

```
formula_sample =  sample ~ Disttoroadm + Populationm
formula_presence = species ~ Elevationm + Precipitationm


## Build dataframes, S <- Sample, P <- Presence
S <- model.frame(formula_sample, DataFrame,na.action='na.pass')
P <- model.frame(formula_presence, DataFrame,na.action='na.pass')

## Split response variables (lefthand side) and covariates (righthand side) from the de
SX <- select(S, -c(1))
PX <- select(P, -c(1))
Sy <- select(S, c(1))
Py <- select(P, c(1))

## Assign names to columns
names(Sy) <- 'response'
names(Py) <- names(Sy)

## Stack both processes into same dataframe
## First the responses (Y) will be concatenated by row.
Y = rbind(Sy,Py)
```

To join both sets of covariates (SX and PX) into the stacked design matrix, we need to join the columns of SX and PX. As there's no information of environmental variables for process S and, viceversa, no information of the sampling covariates for process P, we assign 0 to these elements.

For doing this, we first define two 2x2 matrices and perform the kronnecker product to generate a block diagonal design matrix to be used as the covariates of the stacked design matrix.

```
T1 <- matrix(rep(0,4), ncol = 2)
T2 <- matrix(rep(0,4), ncol = 2)
T1[1,1] <- 1
T2[2,2] <- 1

## Perform Kronnecker with different covariates (Block diagonal)
X <- data.frame((T1 %x% as.matrix(SX)) + (T2 %x% as.matrix(PX)))
```

```
names(X) <- c(names(SX),names(PX))

## Lastly, we bind the stacked response variable with the stacked covariance matrix.
SDM <- cbind(Y,X)
```

2.2 Defining other necessary arguments

```
## Get number of elements in the lattice,
nK <- dim(M_bis)[1]

## make sequence vector for id.area
## This is possible because the rows in adjacency matrix M_bis preserve the order of ea
ida <- data.frame(seq(nK))
idarea <- unlist(rbind(ida,ida))
## Assign values for the unstructured random effect (Zx, Zy).
## In this case, all elements of S share the same random effect Zx.
## Similarly, all elements of P share the same random effect Zy.
## This is done with
Zx <- rep(x = 1,times = nK)
Zy <- rep(x = 2,times = nK)
## Stack Zx and Zy to create a vector of labels used to define
## the unstructured random effect.
indre <- c(Zx,Zy)
```

2.3 Running the model

```
#formula_sample =  sample ~ Disttoroadm + Populationm
#formula_presence = species ~ Elevationm + Precipitationm
formula <- response ~ Disttoroadm + Populationm
                      + Elevationm + Precipitationm
ind.re = c(rep(1,nK),rep(2,nK))

ntot <- dim(SDM)[1]
trials <- rep(1,ntot)
## uncomment for real testing
#burnin = 1000000
#n.sample = 1200000
#thin = 50
## Toy example
burnin = 100
```

```
n.sample = 120
thin = 1

model2 <- S.CARmultilevel(formula,family = 'binomial',
                          trials=as.numeric(trials),
                          W=M_bis,
                          ind.area=idarea,
                          #ind.re=factor(idarea),
                          ind.re = factor(ind.re),
                          rho=1,
                          burnin=burnin,
                          n.sample=n.sample,
                          data=SDM)
```

Similarly to model I, the posterior summary statistics can be inspected by accessing the attribute summary.results.

```
model2$P$summary.results
```

3 Model III

This model assumes that processes S and P are correlated to each other. Both of them have a specific Gaussian Markov random field and these two spatial structures are correlated. The correlated spatial structure follows a multivariate CAR specification ([8, 4]). The structure of the model is visualised as a directed acyclic graph (DAG) in the figure 3. For a detail explanation of the model refer to the supplementary materials of the manuscript. The joint probability distribution of P and S is given by:

$$[P, S] = [P|GMRF_p][S|GMRF_s]$$

3.1 Running the model

Applying this model is straight forward. We only need to define an appropriate formula object.

```
formula <- cbind(species,sample) ~ Elevationm + Precipitationm +
                                   Disttoroadm + Populationm
## Get number areal elements
```

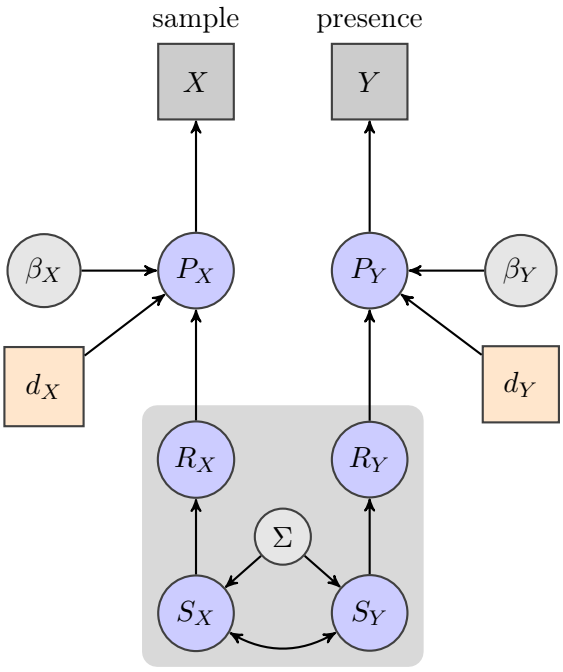

Figure 3: DAG for model III

```
K <- dim(M_bis)[1]
## Calculate new trial vector
trials <- matrix(rep(1.0,K*2), ncol=2)

## Toy example, increase n.sample and burnin significantly for real applications.
burnin = 400
n.sample = 500
thin = 5
model3 <- MVS.CARleroux(formula ,
                        family = 'binomial',
                        trials=trials,
                        W=M_bis,
                        rho = 1,
                        burnin = burnin,
                        n.sample = n.sample,
                        data = DataFrame
                         )
```

To examine the model diagnostics use modelfit, as usual.

```
model3$modelfit
```

DIC	p.d	WAIC	p.w	LMPL	loglikelihood
3976.2	93.21	3966.76	81.39	-1982.01	-1894.88

:Logbook:

4 Extracting posterior sampling and exporting to standard formats

In the manuscript, the visualisations were performed with Python/Matplotlib/pyvis. Showing how to plot results are outside of the scope of the present guide. Nevertheless, we present here a procedure to extract summary statistics of the posterior sample for using (e.g. plotting) using other software.

:END:

References

[1] Juan Manuel Escamilla Mólgora, Luigi Sedda, Peter Diggle, and Peter Atkinson. A joint distribution framework to improve presence-only species distribution models by exploiting citizen sampling effort. 2021.

[2] P. J. Diggle, J. A. Tawn, and R. A. Moyeed. Model-based geostatistics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(3):299–350, jan 2002.

[3] Julian Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192–236, 1974.

[4] Duncan Lee. CARBayes : An R Package for Bayesian Spatial Modeling with Conditional Autoregressive Priors. *Journal of Statistical Software*, 55(13):1–24, nov 2013.

[5] GBIF.org. GBIF Occurrence Download, 2016.

[6] Håvard Rue and Leonhard Held. *Gaussian markov random fields: Theory and applications*. Chapman and Hall/CRC, 2005.

[7] Seymour Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

[8] Leo Kavanagh, Duncan Lee, and Gwilym Pryce. Is Poverty Decentralizing? Quantifying Uncertainty in the Decentralization of Urban Poverty. *Annals of the American Association of Geographers*, 106(6):1286–1298, nov 2016.