Using integrated multispecies occupancy models to map co-occurrence between bottlenose dolphins and fisheries in the Gulf of Lion, French Mediterranean Sea.

R codes

Valentin Lauret, Hélène Labach, Léa David, Matthieu Authier, Olivier Gimenez

Table of Contents

# Load required packages

library(tidyverse)  
library(sf)  
library(nimble)

# Load data

load("IMSOdata.rdata")

# IMSO

Get the ingredients for GAMs using package jagam developed by Simon Wood and basically hacks what is built by the package mgcv.

yy\_dolphin <- apply(y, 1, max, na.rm = TRUE)  
yy\_dolphin[yy\_dolphin ==1] <- 0  
yy\_dolphin[yy\_dolphin >1] <- 1  
  
library(mgcv)  
res <- jagam(yy\_dolphin ~ s(stbathy) + s(coordx, coordy, bs = "gp"),   
 family = "binomial",  
 file = "psi.txt")

We do it in NIMBLE because the MCMC run is faster than that of JAGS.

## BUGS model

IMSO <- nimbleCode({   
   
 ## state process  
 for(j in 1:nsite){  
 z[j] ~ dcat(psi[j, 1:4])  
 }  
 # occupancy probabilities  
   
 psi[1:nsite, 1] <- 1 / (1 + sum(prop[1:nsite, 1:3])) # unoccupied  
 psi[1:nsite, 2] <- prop[1:nsite, 1] / (1 + sum(prop[1:nsite, 1:3])) # occupied by species A and not B  
 psi[1:nsite, 3] <- prop[1:nsite, 2] / (1 + sum(prop[1:nsite, 1:3])) # occupied by species B and not A  
 psi[1:nsite, 4] <- prop[1:nsite, 3] / (1 + sum(prop[1:nsite, 1:3])) # occupied by both species A and B  
   
 ## observation process  
 for(j in 1:nsite) {  
 for(k in 1:nyear) {  
 y[j, k] ~ dcat(obs[j, k, 1:16, z[j]])  
 }  
 }  
   
 # detection matrix with obs for observations and state = true states  
 # oSee supplementary information for details  
 # given state = unoccupied,  
 for(j in 1:nsite) {  
 for(k in 1:nyear) {  
 # state 1 = no species use the site  
 obs[j, k, 1, 1] <- 1 # prob obs = 1   
 obs[j, k, 2, 1] <- 0 # prob obs = 2  
 obs[j, k, 3, 1] <- 0 # prob obs = 3  
 obs[j, k, 4, 1] <- 0 # prob obs = 4  
 obs[j, k, 5, 1] <- 0 # prob obs = 5   
 obs[j, k, 6, 1] <- 0 # prob obs = 6  
 obs[j, k, 7, 1] <- 0 # prob obs = 7  
 obs[j, k, 8, 1] <- 0 # prob obs = 8  
 obs[j, k, 9, 1] <- 0 # prob obs = 9   
 obs[j, k, 10, 1] <- 0 # prob obs = 10  
 obs[j, k, 11, 1] <- 0 # prob obs = 11  
 obs[j, k, 12, 1] <- 0 # prob obs = 12  
 obs[j, k, 13, 1] <- 0 # prob obs = 13   
 obs[j, k, 14, 1] <- 0 # prob obs = 14  
 obs[j, k, 15, 1] <- 0 # prob obs = 15  
 obs[j, k, 16, 1] <- 0 # prob obs = 16  
   
 # given state 2 = occupied by species A and not B,  
 obs[j, k, 1, 2] <- 1 - pAg[j,k] - pAs[j,k] + pAg[j,k] \* pAs[j,k] # prob obs = 1   
 obs[j, k, 2, 2] <- pAg[j,k] \* (1 - pAs[j,k]) # prob obs = 2  
 obs[j, k, 3, 2] <- 0 # prob obs = 3  
 obs[j, k, 4, 2] <- 0 # prob obs = 4  
 obs[j, k, 5, 2] <- pAs[j,k] \* (1 - pAg[j,k]) # prob obs = 5   
 obs[j, k, 6, 2] <- pAs[j,k] \* pAg[j,k] # prob obs = 6  
 obs[j, k, 7, 2] <- 0 # prob obs = 7  
 obs[j, k, 8, 2] <- 0 # prob obs = 8  
 obs[j, k, 9, 2] <- 0 # prob obs = 9   
 obs[j, k, 10, 2] <- 0 # prob obs = 10  
 obs[j, k, 11, 2] <- 0 # prob obs = 11  
 obs[j, k, 12, 2] <- 0 # prob obs = 12  
 obs[j, k, 13, 2] <- 0 # prob obs = 13   
 obs[j, k, 14, 2] <- 0 # prob obs = 14  
 obs[j, k, 15, 2] <- 0 # prob obs = 15  
 obs[j, k, 16, 2] <- 0 # prob obs = 16  
   
 # given state 3 = occupied by species B and not A,  
 obs[j, k, 1, 3] <- 1 - pBg[j,k] - pBs[j,k] + pBg[j,k] \* pBs[j,k] # prob obs = 1   
 obs[j, k, 2, 3] <- 0 # prob obs = 2  
 obs[j, k, 3, 3] <- pBg[j,k] \* (1 - pBs[j,k]) # prob obs = 3  
 obs[j, k, 4, 3] <- 0 # prob obs = 4  
 obs[j, k, 5, 3] <- 0 # prob obs = 5   
 obs[j, k, 6, 3] <- 0 # prob obs = 6  
 obs[j, k, 7, 3] <- 0 # prob obs = 7  
 obs[j, k, 8, 3] <- 0 # prob obs = 8  
 obs[j, k, 9, 3] <- pBs[j,k] \* (1 - pBg[j,k]) # prob obs = 9   
 obs[j, k, 10, 3] <- 0 # prob obs = 10  
 obs[j, k, 11, 3] <- pBs[j,k] \* pBg[j,k] # prob obs = 11  
 obs[j, k, 12, 3] <- 0 # prob obs = 12  
 obs[j, k, 13, 3] <- 0 # prob obs = 13   
 obs[j, k, 14, 3] <- 0 # prob obs = 14  
 obs[j, k, 15, 3] <- 0 # prob obs = 15  
 obs[j, k, 16, 3] <- 0 # prob obs = 16  
   
 # given state 4 = occupied by both species B and A,  
 obs[j, k, 1, 4] <- (1 - pAs[j,k]) \* (1 - pAg[j,k]) \* (1- pBs[j,k]) \* (1 - pBg[j,k]) # prob obs = 1   
 obs[j, k, 2, 4] <- (1 - pAs[j,k]) \* (1 - pBs[j,k]) \* pAg[j,k] \* (1 - pBg[j,k]) # prob obs = 2  
 obs[j, k, 3, 4] <- (1 - pAs[j,k]) \* (1 - pBs[j,k]) \* pBg[j,k] \* (1 - pAg[j,k]) # prob obs = 3  
 obs[j, k, 4, 4] <- (1 - pAs[j,k]) \* (1 - pBs[j,k]) \* pAg[j,k] \* pBg[j,k] # prob obs = 4  
 obs[j, k, 5, 4] <- pAs[j,k]\*(1 - pBs[j,k]) \* (1 - pAg[j,k]) \* (1 - pBg[j,k]) # prob obs = 5   
 obs[j, k, 6, 4] <- pAs[j,k]\*(1 - pBs[j,k]) \* pAg[j,k] \* (1 - pBg[j,k]) # prob obs = 6  
 obs[j, k, 7, 4] <- pAs[j,k]\*(1 - pBs[j,k]) \* pBg[j,k] \* (1 - pAg[j,k]) # prob obs = 7  
 obs[j, k, 8, 4] <- pAs[j,k]\*(1 - pBs[j,k]) \* pAg[j,k] \* pBg[j,k] # prob obs = 8  
 obs[j, k, 9, 4] <- pBs[j,k]\*(1 - pAs[j,k]) \* (1 - pAg[j,k]) \* (1 - pBg[j,k]) # prob obs = 9   
 obs[j, k, 10, 4] <- pBs[j,k]\*(1 - pAs[j,k]) \* pAg[j,k] \* (1 - pBg[j,k]) # prob obs = 10  
 obs[j, k, 11, 4] <- pBs[j,k]\*(1 - pAs[j,k]) \* pBg[j,k] \* (1 - pAg[j,k]) # prob obs = 11  
 obs[j, k, 12, 4] <- pBs[j,k]\*(1 - pAs[j,k]) \* pAg[j,k] \* pBg[j,k] # prob obs = 12  
 obs[j, k, 13, 4] <- pAs[j,k] \* pBs[j,k] \* (1 - pAg[j,k]) \* (1 - pBg[j,k]) # prob obs = 13   
 obs[j, k, 14, 4] <- pAs[j,k] \* pBs[j,k] \* pAg[j,k] \* (1 - pBg[j,k]) # prob obs = 14  
 obs[j, k, 15, 4] <- pAs[j,k] \* pBs[j,k] \* pBg[j,k] \* (1 - pAg[j,k]) # prob obs = 15  
 obs[j, k, 16, 4] <- pAs[j,k] \* pAg[j,k] \* pBs[j,k] \* pBg[j,k] # prob obs = 16  
 }  
 }   
 ## priors for...  
 # occupancy probabilities  
   
 for(j in 1:nsite) {  
 log(prop[j, 1]) <- theta1[j]   
 log(prop[j, 2]) <- theta2[j]   
 log(prop[j, 3]) <- theta3[j]  
 }  
   
 theta1[1:nsite] <- X[1:nsite,1:42] %\*% b1[1:42] ## linear predictor  
 theta2[1:nsite] <- X[1:nsite,1:42] %\*% b2[1:42] ## linear predictor  
 theta3[1:nsite] <- X[1:nsite,1:42] %\*% b3[1:42] ## linear predictor  
   
 b1[1] ~ dnorm(0,0.01)   
 b2[1] ~ dnorm(0,0.01)   
 b3[1] ~ dnorm(0,0.01)   
   
 ## prior for s(bathy)   
 K11[1:9,1:9] <- S1[1:9,1:9] \* lambda[1, 1] + S1[1:9,10:18] \* lambda[2, 1]  
 K12[1:9,1:9] <- S1[1:9,1:9] \* lambda[1, 2] + S1[1:9,10:18] \* lambda[2, 2]  
 K13[1:9,1:9] <- S1[1:9,1:9] \* lambda[1, 3] + S1[1:9,10:18] \* lambda[2, 3]  
 b1[2:10] ~ dmnorm(zero[2:10], K11[1:9,1:9])   
 b2[2:10] ~ dmnorm(zero[2:10], K12[1:9,1:9])   
 b3[2:10] ~ dmnorm(zero[2:10], K13[1:9,1:9])   
   
 ## prior for s(coordx,coordy)   
 K21[1:32,1:32] <- S2[1:32,1:32] \* lambda[3, 1] + S2[1:32,33:64] \* lambda[4, 1]  
 K22[1:32,1:32] <- S2[1:32,1:32] \* lambda[3, 2] + S2[1:32,33:64] \* lambda[4, 2]  
 K23[1:32,1:32] <- S2[1:32,1:32] \* lambda[3, 3] + S2[1:32,33:64] \* lambda[4, 3]  
 b1[11:42] ~ dmnorm(zero[11:42], K21[1:32,1:32])   
 b2[11:42] ~ dmnorm(zero[11:42], K22[1:32,1:32])   
 b3[11:42] ~ dmnorm(zero[11:42], K23[1:32,1:32])   
   
 ## smoothing parameter priors  
 for (i in 1:4) {  
 for (kk in 1:3){  
 lambda[i, kk] ~ dgamma(.05,.005)  
 rho[i, kk] <- log(lambda[i, kk])  
 }  
 }  
 # detection probabilities  
 # VL: There are four detections probabilities now pAs, pAg, pBg, pBs  
 for(j in 1:nsite) {  
 for(k in 1:nyear) {  
 pAs[j, k] <- (1/(1 + exp(-(beta[1] + beta[2] \* effS[j, k]))))\*effindS[j,k]   
   
 pBs[j, k] <- (1/(1 + exp(-(beta[3] + beta[4] \* effS[j, k]))))\*effindS[j,k]   
   
 pAg[j, k] <- (1/(1 + exp(-(beta[5] + beta[6] \* effG[j, k])))) \*effindG[j,k]   
   
 pBg[j, k] <- (1 /(1+ exp(-(beta[7] + beta[8] \* effG[j, k]))))\*effindG[j,k]   
   
 }  
 }  
 for (i in 1:8){  
 beta[i] ~ dnorm(0,1)  
 }  
})

## Bundle data

Specify data, initial values, parameters to be monitored and various MCMC details:

## Build, compile and run model with NIMBLE

Rmodel <- nimbleModel(IMSO, constants, data, inits)  
Rmodel$initializeInfo()  
Rmodel$calculate()  
  
conf <- configureMCMC(Rmodel)  
conf$printMonitors()   
conf$addMonitors("z")  
  
conf$printSamplers(byType= TRUE)  
  
# Build and compile MCMC  
Rmcmc <- buildMCMC(conf)  
Cmodel <- compileNimble(Rmodel)  
Cmcmc <- compileNimble(Rmcmc, project = Cmodel)  
  
t <- system.time(samples2 <- runMCMC(Cmcmc, niter = 100000, nburnin = 10000, nchains = 3,  
 samplesAsCodaMCMC = TRUE))