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

R codes and data to implement integrated multispecies occupancy models

Valentin Lauret[1], Hélène Labach[2], Léa David[3], Matthieu Authier[4,5], Olivier Gimenez[1]

1 Load required packages

```r
library(tidyverse)
library(sf)
library(nimble)
```

2 Load data

```r
load("IMSOdata.rdata")
```

In the input data, we have:

- `y` is the detection / no-detection dataset that stores values ranging from 1 to 16 corresponding to detection status in each of the 279 grid-cells (in rows) and each of the 4 sampling occasion (in columns).
- `seffG` and `seffS` that store the sampling effort covariates of GDEGeM boat surveys and SAMM aerial surveys respectively. Values correspond to the **scaled** length (in km) of the survey in each of the 279 grid-cells (in rows) and each of the 4 sampling occasion (in columns). `effindG` and `effindS` store binary 0/1 values indicating if each of the 279 grid-cells (in rows) and each of the 4 sampling occasion (in columns) have been sampled by GDEGeM boat surveys and SAMM aerial surveys respectively.
- `stbathy` stores the **scaled** depth covariate in each of the 279 grid-cells.

3 IMSO

3.1 Get the variables of the integrated multispecies occupancy model (IMSO)

- `z[j]` is the variable describing the latent occupancy state at grid-cell `j`.
- `y[j,k]` is the variable describing the observation detection / no-detection data at grid-cell `j` and sampling occasion `k`.
- `psi[j,k]` is the occupancy probability at grid-cell `j` for sampling occasion `k`.
- `obs[j,k, 1:16, z[j]]` is th vector of the detection probabilities of each possible observation `y[j,k]` at grid-cell `j` for sampling occasion `k`, conditional on `z[j]`.
- `pAs[j,k]`, `pAg[j,k]`, `pBs[j,k]`, `pBg[j,k]` are the detection probabilities of bottlenose dolphins by aerial surveys, of bottlenose dolphins by boat surveys, of fishing trawlers by aerial surveys, and of fishing trawlers by boat surveys respectively at grid-cell `j` and sampling occasion `k`.
- `beta[1:8]` are the intercepts and slope assoicated with the logistic regressions that estimate detection probabilities `pAs[j,k]`, `pAg[j,k]`, `pBs[j,k]`, and `pBg[j,k]`.

- lambda, b1, b2, b3, X , S1, zero are GAM parameters used to parametrize GAMs in BUGS language, (for additional details one can refer to this example by Mason Fidino using {mgcv} R package to implement a GAM in an occupancy model).

3.2 R codes

Get the ingredients for GAMs using package jagam developed by Simon Wood and basically hacks what is built by the package mgcv. We implemented the GAM with default K=10 knots and bs = "gp" Gaussian process smooths for penalized splines.

```
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 ~ 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.

3.3 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,1:34] %*% b1[1:34] ## linear predictor
  theta2[1:nsite] <- X[1:nsite,1:34] %*% b2[1:34] ## linear predictor
  theta3[1:nsite] <- X[1:nsite,1:34] %*% b3[1:34] ## linear predictor


  b1[1] ~ dnorm(0,0.01)
  b2[1] ~ dnorm(0,0.01)
  b3[1] ~ dnorm(0,0.01)

  ## prior for s(bathy)
  b1[2] ~ dnorm(0,0.01)
  b2[2] ~ dnorm(0,0.01)
  b3[2] ~ dnorm(0,0.01)

  ## prior for s(coordx,coordy)
  K21[1:32,1:32] <- S1[1:32,1:32] * lambda[1, 1]  + S1[1:32,33:64] * lambda[2, 1]
  K22[1:32,1:32] <- S1[1:32,1:32] * lambda[1, 2]  + S1[1:32,33:64] * lambda[2, 2]
  K23[1:32,1:32] <- S1[1:32,1:32] * lambda[1, 3]  + S1[1:32,33:64] * lambda[2, 3]
  b1[3:34] ~ dmnorm(zero[1:32], K21[1:32,1:32])
  b2[3:34] ~ dmnorm(zero[1:32], K22[1:32,1:32])
  b3[3:34] ~ dmnorm(zero[1:32], K23[1:32,1:32])

  ## smoothing parameter priors
  for (i in 1:2) {
    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)
  }
})
```

3.4 Bundle data

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

3.5 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))
```