ajustement modèles à 2 espèces sur données GDEGeM sur Golfe du Lion

Olivier Gimenez

10/13/2020

Format and visualise data

Load grid and occupancy data.

```
load("pays.rdata")
grid <- st_read("Grid/grid.shp") %>% st_transform(crs = st_crs(pays))
```

```
## Reading layer ‘grid’ from data source ‘/Users/oliviergimenez/Dropbox/Mon Mac (MacBook-Pro-de-Olivier
## Simple feature collection with 4356 features and 3 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 701000 ymin: 5886622 xmax: 1467639 ymax: 6390000
## projected CRS:  Lambert_Conformal_Conic
```

```
# focus Golfe du Lion
grid <-  grid %>% st_crop(xmin = 700000, xmax = 900000, ymin = 6140000, ymax = 6300000)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

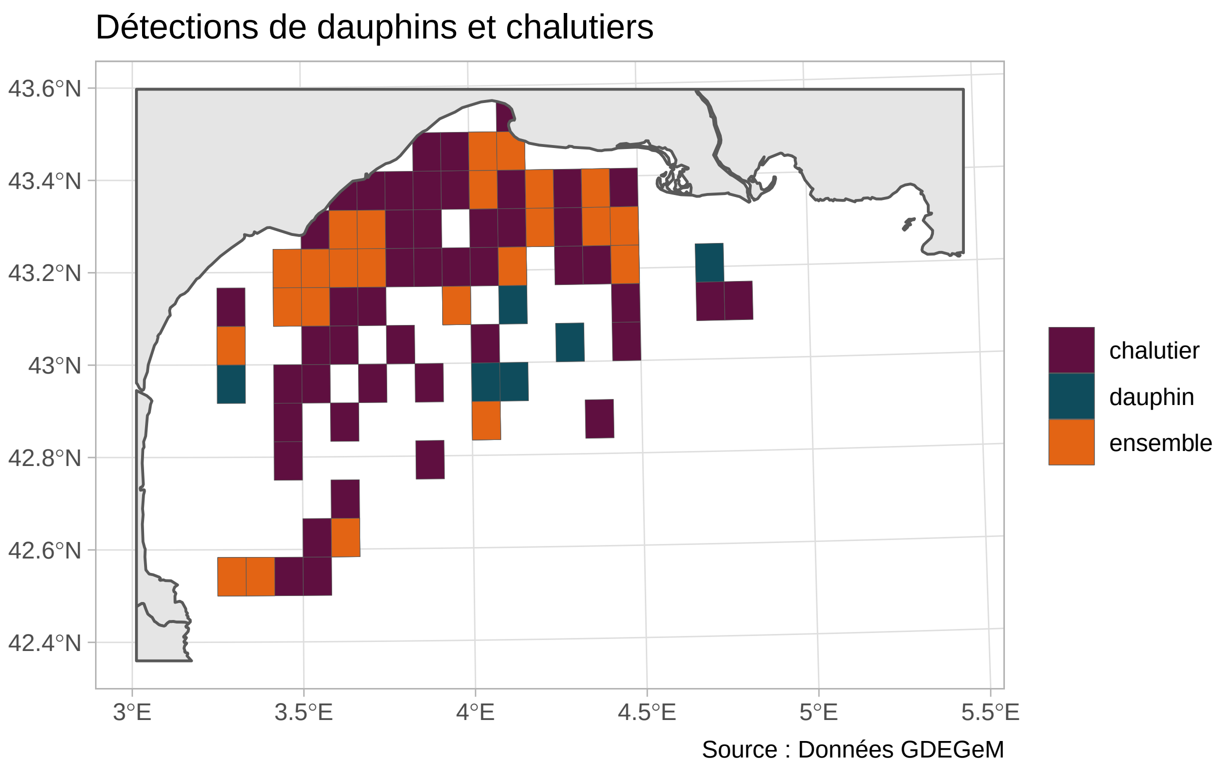
```
pays <- pays %>% st_crop(st_bbox(grid))
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
load('msoccu_gd.rdata')
```

Visualise data.

```
df <- bind_rows(multioccu_gd$dauphins %>% add_column(species = "dauphin"),
          multioccu_gd$chalut %>% add_column(species = "chalutier"))
cooc <- which(multioccu_gd$chalut$obs == 1 & multioccu_gd$dauphins$obs == 1)

df %>%
  ggplot() +
  geom_sf(data = df %>% filter(obs == 1, species == "dauphin"), lwd = 0.1, aes(fill = "dauphin")) +
  geom_sf(data = df %>% filter(obs == 1, species == "chalutier"), lwd = 0.1, aes(fill = "chalutier")) +
  geom_sf(data = grid %>% slice(cooc), lwd = 0.1, aes(fill = "ensemble"))+
  scale_fill_manual(name = "",
                    values = c("dauphin" = "#0F4C5C",
                               "chalutier" = "#5f0f40",
                               "ensemble" = "#e36414")) +
  geom_sf(data = pays) +
  labs(title = "Détections de dauphins et chalutiers",
       caption = "Source : Données GDEGeM")
```



Build datasets.

```
# sampling effort
effort <- multioccu_gd$effort %>%
  select(autumn:summer) %>%
  as_tibble() %>%
  select(-geometry) %>%
  as.matrix() # 397 x 4

# dolphin detections/non-detections
y_dolphin <- df %>%
  filter(species == "dauphin") %>%
  select(autumn:summer) %>%
  as_tibble() %>%
  select(-geometry) %>%
  as.matrix()
y_dolphin[effort == 0] <- NA
ind_dolphin <- apply(y_dolphin, 1, function(x) all(is.na(x)))
y_dolphin <- y_dolphin[ !ind_dolphin, ]
```

```r
# fishing boats detections/non-detections
y_fishing <- df %>%
  filter(species == "chalutier") %>%
  select(autumn:summer) %>%
  as_tibble() %>%
  select(-geometry) %>%
  as.matrix()
y_fishing[effort == 0] <- NA
ind_fishing <- apply(y_fishing, 1, function(x) all(is.na(x)))
y_fishing <- y_fishing[ !ind_fishing, ]

# grid cells coordinates
coord <- df %>%
  filter(species == "dauphin") %>%
  select(autumn:summer) %>%
  st_centroid() %>%
  st_coordinates() %>%
  as_tibble() %>%
  mutate(easting = (X - mean(X)) / sd(X),
         northing = (Y - mean(Y)) / sd(Y)) %>%
  select(easting, northing) %>%
  as.matrix()
```

```
## Warning in st_centroid.sf(.): st_centroid assumes attributes are constant over
## geometries of x
```

```r
mask <- apply(effort == 0, 1, sum) == 4
coord <- coord[!mask,]
dim(coord)
```

```
## [1] 148   2
```

```r
# means and standard deviations used to standardise the grid cells coordinates
temp <- df %>%
  filter(species == "dauphin") %>%
  st_centroid() %>%
  st_coordinates() %>%
  as.matrix()
```

```
## Warning in st_centroid.sf(.): st_centroid assumes attributes are constant over
## geometries of x
```

```r
meanX <- mean(temp[,1])
sdX <- sd(temp[,1])
meanY <- mean(temp[,2])
sdY <- sd(temp[,2])
```

An ounce of theory

We consider a two-species static occupancy model à la Rota et al. (2016).

Ignoring the site index, we use the following notation for the occupancy probabilities:

- ψ_{11} is the prob. that species 1 and species 2 are both present;

- ψ_{10} is the prob. that species 1 is present and species 2 is absent;
- ψ_{01} is the prob. that species 1 is absent and species 2 is present;
- ψ_{00} is the prob. that species 1 and species 2 are both absent, with avec $\psi_{11} + \psi_{10} + \psi_{01} + \psi_{00} = 1$.

The marginal probabilities of occupancy are:

- $\Pr(z_1 = 1) = \Pr(\text{species 1 is present}) = \psi_{10} + \psi_{11}$
- $\Pr(z_2 = 1) = \Pr(\text{species 2 is present}) = \psi_{01} + \psi_{11}$
- $\Pr(z_1 = 0) = \Pr(\text{species 1 is absent}) = \psi_{01} + \psi_{00}$
- $\Pr(z_2 = 0) = \Pr(\text{species 2 is absent}) = \psi_{10} + \psi_{00}$

And the conditional probabilities (reminder: $\Pr(\text{A}|\text{B}) = \Pr(\text{A and B})/\Pr(\text{B})$):

- $\Pr(z_1 = 1|z_2 = 0) = \psi_{10}/(\psi_{10} + \psi_{00}) = \Pr(\text{species 1 is present given species 2 is absent})$;
- $\Pr(z_1 = 1|z_2 = 1) = \psi_{11}/(\psi_{11} + \psi_{01}) = \Pr(\text{species 1 is present given species 2 is present})$;
- $\Pr(z_2 = 1|z_1 = 0) = \psi_{01}/(\psi_{01} + \psi_{00}) = \Pr(\text{species 2 is present given species 1 is absent})$;
- $\Pr(z_2 = 1|z_1 = 1) = \psi_{11}/(\psi_{11} + \psi_{10}) = \Pr(\text{species 2 is present given species 1 is present})$.

It is important to note that the function `occuMulti` in `unmarked` doesn't work directly on the occupancy probabilities but on the so-called natural parameters (in that specific order):

- $f_1 = \log(\psi_{10}/\psi_{00})$;
- $f_2 = \log(\psi_{01}/\psi_{00})$;
- $f_{12} = \log(\psi_{00}\psi_{11}/\psi_{10}\psi_{01})$,

that is:

- $\psi_{11} = \exp(f_1 + f_2 + f_{12})/\text{den}$;
- $\psi_{10} = \exp(f_1)/\text{den}$;
- $\psi_{01} = \exp(f_2)/\text{den}$, where $\text{den} = 1 + \exp(f_1) + \exp(f_2) + \exp(f_1 + f_2 + f_{12})$:

Analyses with `unmarked`

Load `unmarked` awesome package.

```r
library(unmarked)
```

```
## Loading required package: lattice
```

Format data.

```r
y <- list(y_dolphin, y_fishing)
names(y) <- c('dolphin','fishing')

ind_effort <- apply(effort, 1, sum)
cov_effort <- effort[ ind_effort!=0, ]
st_effort <- matrix(scale(cov_effort), ncol = 4)
det_covs <- list()
```

```r
det_covs[[1]] <- st_effort
det_covs[[2]] <- st_effort
names(det_covs) <- paste('det_cov',1:2,sep='')

data <- unmarkedFrameOccuMulti(y = y, obsCovs = det_covs) #,siteCovs=occ_covs,obsCovs=det_covs)
```

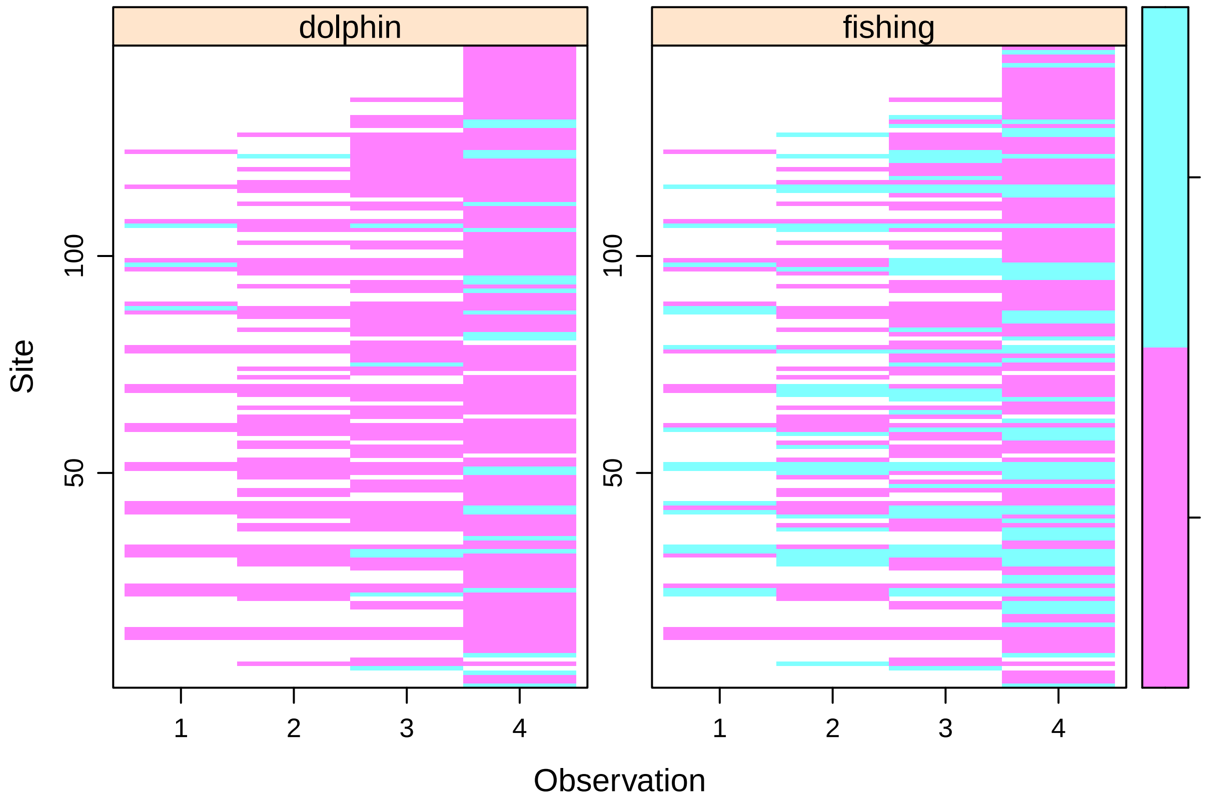
Summary stats.

```r
summary(data)
```

```
## unmarkedFrame Object
##
## 148 sites
## 2 species: dolphin fishing
## Maximum number of observations per site: 4
## Mean number of observations per site:
## dolphin: 2.2027027027027  fishing: 2.2027027027027
## Sites with at least one detection:
## dolphin: 29  fishing: 71
## Tabulation of y observations:
## dolphin:
##    0    1 <NA>
##  294   32  266
## fishing:
##    0    1 <NA>
##  204  122  266
##
## Observation-level covariates:
##     det_cov1          det_cov2
##  Min.   :-1.0780   Min.   :-1.0780
##  1st Qu.:-0.5422   1st Qu.:-0.5422
##  Median :-0.3815   Median :-0.3815
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.1937   3rd Qu.: 0.1937
##  Max.   : 5.1519   Max.   : 5.1519
```

Visualize.

```r
plot(data)
```



Specific effects on parameters. Natural parameters are constant, detection probabilities are species-species, and a function of the sampling effort.

```
occFormulas <- c('~1','~1','~1')
detFormulas <- c('~det_cov1','~det_cov2')
```

Fit model.

```
fit <- occuMulti(detFormulas, occFormulas, data)
```

```
## Warning in .local(umf, ...): Missing detections at sites: 1, 2, 3, 4, 5, 6, 7,
## 8, 9, 10, 11, 15, 16, 17, 18, 19, 20, 21, 25, 26, 27, 28, 29, 30, 34, 35, 36,
## 37, 38, 39, 40, 44, 45, 46, 47, 48, 49, 50, 53, 54, 55, 56, 57, 58, 59, 62, 63,
## 64, 65, 66, 67, 68, 71, 72, 73, 74, 75, 76, 77, 80, 81, 82, 83, 84, 85, 86, 89,
## 90, 91, 92, 93, 94, 95, 96, 100, 101, 102, 103, 104, 105, 106, 109, 110, 111,
## 112, 113, 114, 115, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
## 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144,
## 145, 146, 147, 148
```

Inspect results.

```
fit
```

```
##
## Call:
## occuMulti(detformulas = detFormulas, stateformulas = occFormulas,
##     data = data)
```

```
##
## Occupancy:
##                                 Estimate   SE       z P(>|z|)
## [dolphin] (Intercept)              -8.63 57.8 -0.1492   0.881
## [fishing] (Intercept)              -6.29 70.8 -0.0889   0.929
## [dolphin:fishing] (Intercept)      17.09 91.4  0.1870   0.852
##
## Detection:
##                         Estimate    SE     z  P(>|z|)
## [dolphin] (Intercept)     -2.348 0.241 -9.76 1.65e-22
## [dolphin] det_cov1         0.364 0.134  2.72 6.48e-03
## [fishing] (Intercept)     -0.851 0.190 -4.48 7.31e-06
## [fishing] det_cov2         1.181 0.170  6.94 4.02e-12
##
## AIC: 553.0863
```

Get the natural parameter and detection estimates.

```
mle <- fit@opt$par[1:3]
names(mle) <- c('f1','f2','f12')
mle
```

```
##        f1        f2       f12
## -8.626943 -6.294417 17.089955
```

Get the occupancy estimates.

```
den <- 1 + exp(mle['f1'])+exp(mle['f2'])+exp(mle['f1']+mle['f2']+mle['f12'])
(psi11hat <- exp(mle['f1']+mle['f2']+mle['f12'])/den)
```

```
##        f1
## 0.8972072
```

```
(psi10hat <- exp(mle['f1'])/den)
```

```
##           f1
## 1.838443e-05
```

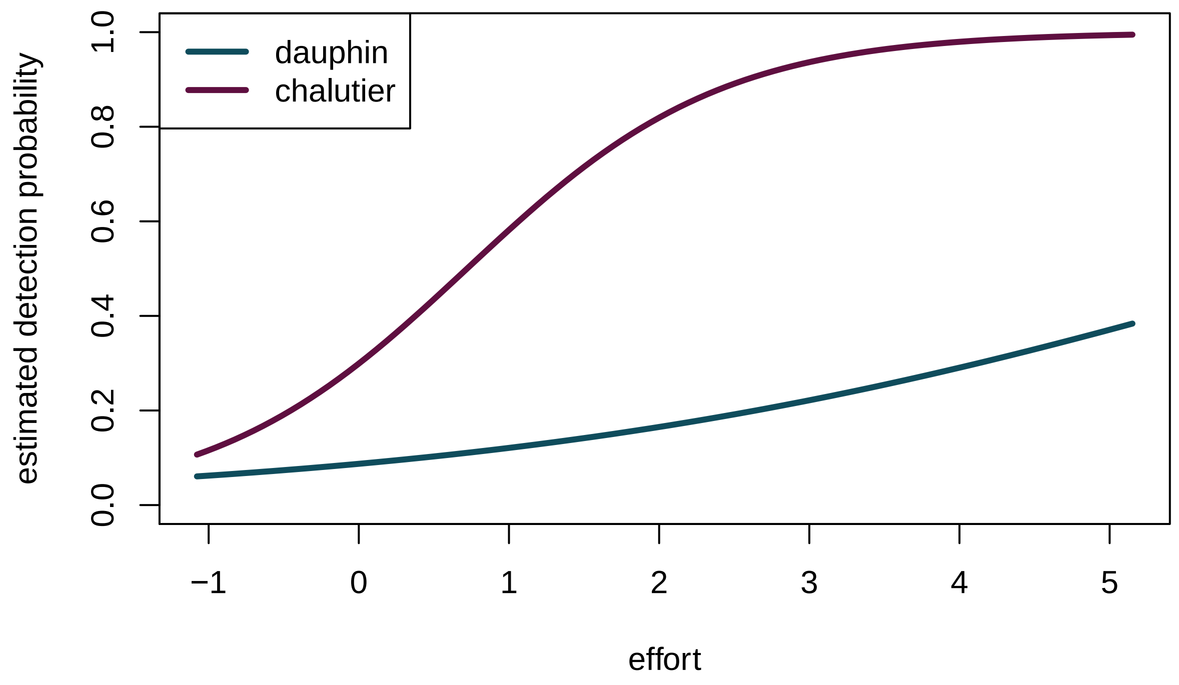
```
(psi01hat <- exp(mle['f2'])/den)
```

```
##          f2
## 0.000189432
```

I do it by hand to understand how `unmarked` works. The easy way is to use `predict(fit,'state')`.

Get the detection estimates.

```
grid_p <- seq(range(st_effort)[1], range(st_effort)[2], length = 100)
logit_p1 <- fit@opt$par[4] + fit@opt$par[5] * grid_p
logit_p2 <- fit@opt$par[6] + fit@opt$par[7] * grid_p
plot(grid_p, plogis(logit_p1),
     type = "l",
     col = "#0F4C5C",
     lwd = 3,
     ylim = c(0,1),
     xlab = "effort",
     ylab = "estimated detection probability")
lines(grid_p, plogis(logit_p2), col = "#5f0f40", lwd = 3)
legend("topleft",
       col = c("#0F4C5C","#5f0f40"),
       lty = c(1,1),
       lwd = 3,
       legend = c("dauphin", "chalutier"))
```



Marginal occupancy.

```
predict(fit,'state',species=1)[1,]
```

```
## Bootstrapping confidence intervals with 100 samples

##   Predicted        SE        lower upper
## 1 0.8972256 0.4902452 4.313518e-51     1
```

```
predict(fit,'state',species=2)[1,]
```

```
## Bootstrapping confidence intervals with 100 samples

##   Predicted        SE       lower upper
## 1 0.8973966 0.4920062 1.00691e-60     1
```

Conditional occupancy.

```
predict(fit,'state',species=1,cond='fishing')[1,] # species 1 | species 2 present
```

```
## Bootstrapping confidence intervals with 100 samples

##   Predicted        SE        lower upper
## 1 0.9997889 0.4996193 1.046167e-93     1
```

```
predict(fit,'state',species=1,cond='-fishing')[1,] # species 1 | species 2 absent
```

```
## Bootstrapping confidence intervals with 100 samples

##      Predicted        SE        lower upper
## 1 0.0001791796 0.4955668 4.618941e-63     1
```

```
predict(fit,'state',species=2,cond='dolphin')[1,] # species 2 | species 1 present
```

```
## Bootstrapping confidence intervals with 100 samples

##   Predicted        SE        lower upper
## 1 0.9999795 0.4986603 1.703042e-77     1
```

```
predict(fit,'state',species=2,cond='-dolphin')[1,] # species 2 | species 1 absent
```

```
## Bootstrapping confidence intervals with 100 samples

##     Predicted        SE        lower upper
## 1 0.001843182 0.4865888 5.589915e-53     1
```

Bayes approach

Let's format the data in a matrix with N rows (sites) and J columns (surveys) with in each cell a 1, 2, 3 or 4 for the observation (or event in the capture-recapture terminology) `none species detected`, `species 1 detected`, `species 2 detected` or `both species detected`.

```
y_unmarked <- y
N <- nrow(y_unmarked[[1]])
J <- ncol(y_unmarked[[1]])
y_jags <- matrix(NA, nrow = N, ncol = J)
for (j in 1:N){
  for (k in 1:J){
    if (is.na(y_unmarked[[1]][j,k])) next # if cell j is not sampled at occasion k, then next
    if (y_unmarked[[1]][j,k] == 0 & y_unmarked[[2]][j,k] == 0) y_jags[j,k] <- 1
    if (y_unmarked[[1]][j,k] == 1 & y_unmarked[[2]][j,k] == 0) y_jags[j,k] <- 2
    if (y_unmarked[[1]][j,k] == 0 & y_unmarked[[2]][j,k] == 1) y_jags[j,k] <- 3
    if (y_unmarked[[1]][j,k] == 1 & y_unmarked[[2]][j,k] == 1) y_jags[j,k] <- 4
  }
}
head(y_jags, 25)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]   NA   NA   NA    4
##  [2,]   NA   NA   NA    1
##  [3,]   NA   NA   NA    1
##  [4,]   NA   NA   NA    2
##  [5,]   NA   NA    4   NA
##  [6,]   NA    3    1    1
##  [7,]   NA   NA    1   NA
##  [8,]   NA   NA   NA    4
##  [9,]   NA   NA   NA    1
## [10,]   NA   NA   NA    1
## [11,]   NA   NA   NA    1
## [12,]    1    1    1    1
## [13,]    1    1    1    1
## [14,]    1    1    1    1
## [15,]   NA   NA   NA    3
## [16,]   NA   NA   NA    1
## [17,]   NA   NA   NA    1
## [18,]   NA   NA   NA    3
## [19,]   NA   NA    1    3
## [20,]   NA   NA    1    3
## [21,]   NA    1   NA    1
## [22,]    3    1    4    3
## [23,]    3    1    3    4
## [24,]    1    1    1    1
## [25,]   NA   NA   NA    3
```

Model w/ constant natural parameters, and detection function of sampling effort plus dolphin detection function of pres/abs of fishing boats

The natural parameters are constant. The detection probabilities for both dolphins and fishing boats depend on the sampling effort (sites and occasions), and the dolphin detection probability is function of the presence/absence of the fishing boats. On the latter, we used the formulation in Waddle et al. 2010 (page 1470) and more precisely:

$$\text{logit}(\Pr(\text{dolphin is detected}|\text{dolphin is present})) = \beta_1 z_{\text{fishing boats}} + \beta_2(1 - z_{\text{fishing boats}}) + \beta_3 \text{sampling effort}$$

Specify model in BUGS language.

```
model <- function() {

  ## state process
  for(j in 1:nsite) {
    z[j] ~ dcat(psi[1:4])
  }

  # occupancy probabilities
  psi[1] <- 1 / (1 + sum(prop[1:3])) # unoccupied
  psi[2] <- prop[1] / (1 + sum(prop[1:3])) # occupied by species A and not B
  psi[3] <- prop[2] / (1 + sum(prop[1:3])) # occupied by species B and not A
  psi[4] <- prop[3] / (1 + sum(prop[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, 1:4, z[j]])
  }
}

# detection matrix with obs for observations and state = true states
# obs take values:
# 1 for none species detected
# 2 for species 1 detected
# 3 for species 2 detected
# 4 for both species detected
# given state = unoccupied,
for(j in 1:nsite) {
  for(k in 1:nyear) {
    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
    # given state = occupied by species A and not B,
    obs[j, k, 1, 2] <- 1 - pA[j, k] # prob obs = 1
    obs[j, k, 2, 2] <- pA[j, k] # prob obs = 2
    obs[j, k, 3, 2] <- 0 # prob obs = 3
    obs[j, k, 4, 2] <- 0 # prob obs = 4
    # given state = occupied by species B and not A,
    obs[j, k, 1, 3] <- 1 - pB[j, k] # prob obs = 1
    obs[j, k, 2, 3] <- 0 # prob obs = 2
    obs[j, k, 3, 3] <- pB[j, k] # prob obs = 3
    obs[j, k, 4, 3] <- 0 # prob obs = 4
    # given state = occupied by both species A and B,
    obs[j, k, 1, 4] <- (1 - pA[j, k]) * (1 - pB[j, k]) # prob obs = 1
    obs[j, k, 2, 4] <- pA[j, k] * (1 - pB[j, k]) # prob obs = 2
    obs[j, k, 3, 4] <- (1 - pA[j, k]) * pB[j, k] # prob obs = 3
    obs[j, k, 4, 4] <- pA[j, k] * pB[j, k] # prob obs = 4
  }
}
## priors for...
# occupancy probabilities
for (i in 1:3){
  log(prop[i]) <- theta[i]
  theta[i] ~ dnorm(0,1)
}
# detection probabilities (pA function of pres/abs of B, as in Waddle et al 2010 page 1470)
for(j in 1:nsite) {
  B_present[j] <- equals(z[j], 3) + equals(z[j], 4)
  for(k in 1:nyear) {
    logit(pA[j, k]) <- beta[1] * B_present[j] + beta[2] * (1 - B_present[j]) + beta[3] * eff[j, k]
    logit(pB[j, k]) <- beta[4] + beta[5] * eff[j, k]
  }
}
for (i in 1:5){
  beta[i] ~ dnorm(0,1)
  }
}
```

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

```
data <- list(y = y_jags,
             nsite = dim(y_jags)[1],
             nyear = dim(y_jags)[2],
             eff = st_effort)
```

Initial values.

```
zinit <- apply(data$y, 1, max, na.rm = TRUE)
zinit[zinit==3] <- 4
inits <- function() {list(z = zinit,
                          beta = rnorm(5, 0, 1),
                          theta = rnorm(3, 0, 1))}
```

Parameters to be monitored.

```
params <- c("prop","theta","beta")
```

MCMC settings

```
ni <- 10000
nb <- 2500
nc <- 2
```

Run Jags from R:

```
library(R2jags)
```

```
## Loading required package: rjags
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
##
## Attaching package: 'R2jags'
```

```
## The following object is masked from 'package:coda':
##
##     traceplot
```

```
ptm <- proc.time()
out <- jags(data = data,
            inits = inits,
            parameters.to.save = params,
            model.file = model,
            n.chains = nc,
            n.iter = ni,
            n.burnin = nb)
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 326
##    Unobserved stochastic nodes: 422
##    Total graph size: 10999
##
## Initializing model
```

```
time <- proc.time() -  ptm
time # 1.5 minutes
```

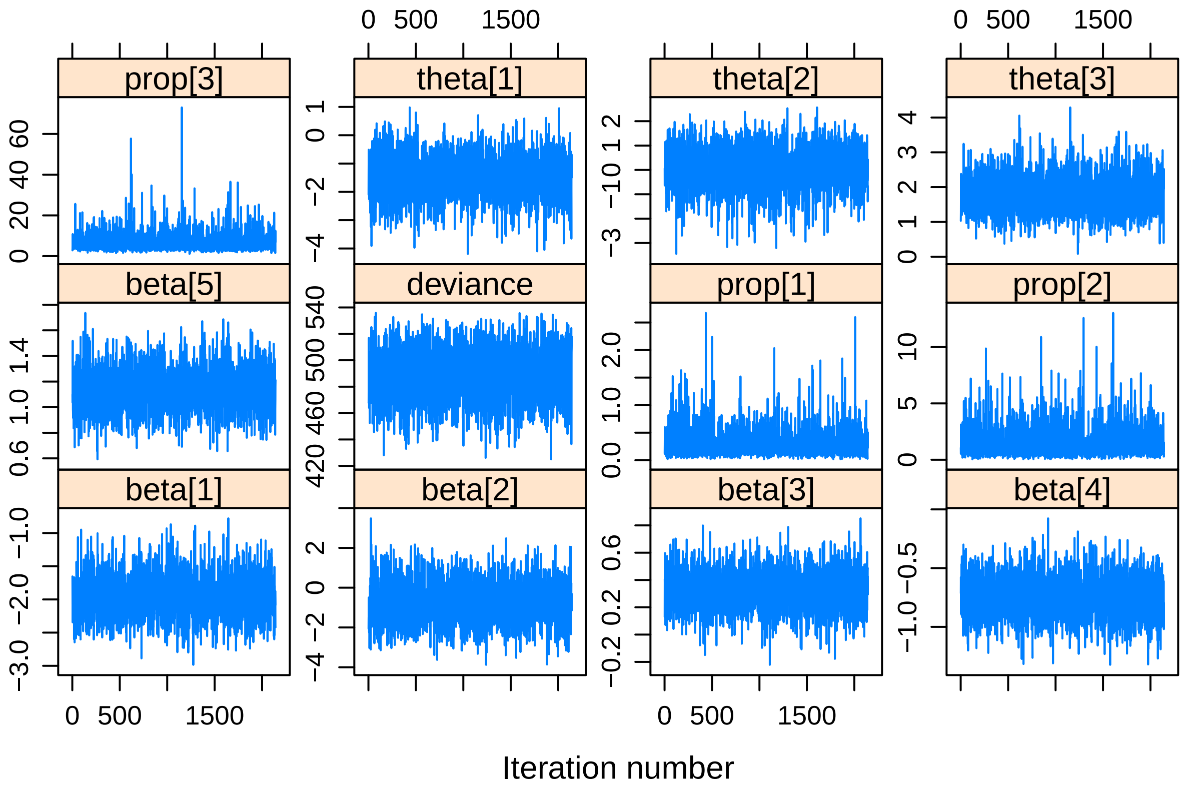
```
##    user  system elapsed
##  77.649   0.090  77.823
```

Save run.

```
save(out, file = "run_gdegem.RData")
```

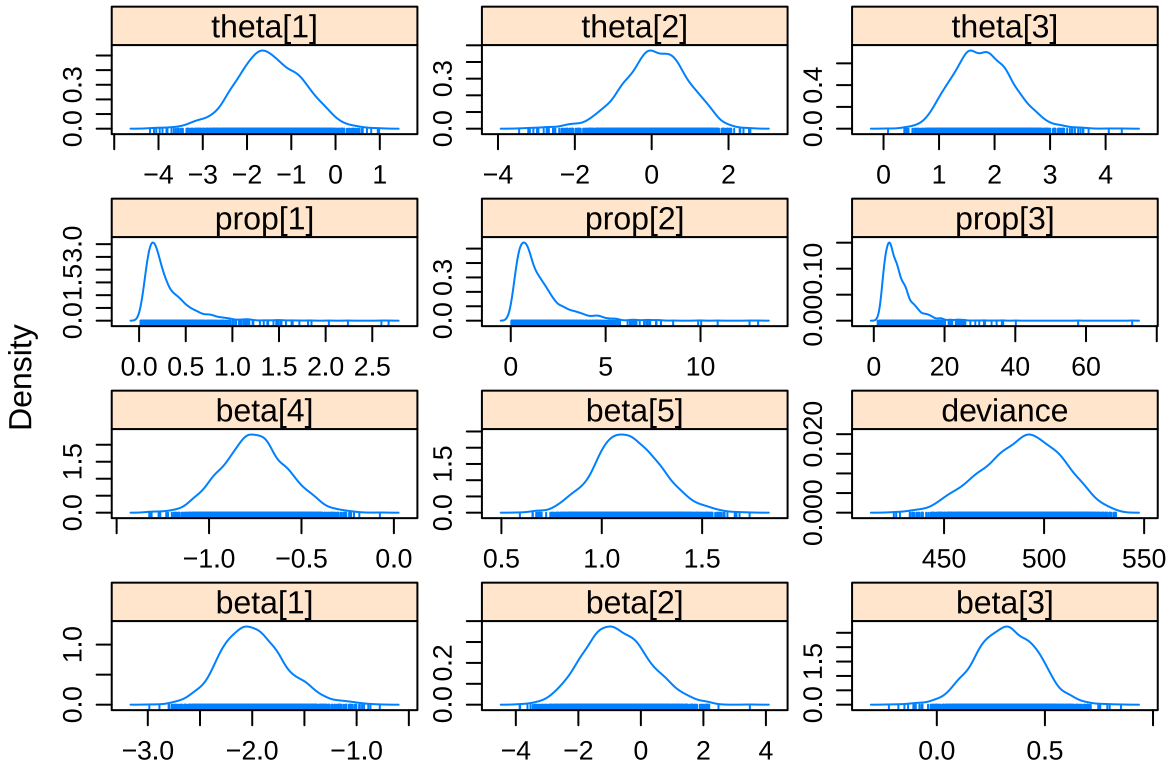
Check convergence.

```
jagsfit.mcmc <- coda::as.mcmc(out$BUGSoutput$sims.matrix)
library(lattice)
xyplot(jagsfit.mcmc, layout = c(4,3))
```



Posterior densities.

```
densityplot(jagsfit.mcmc)
```



Print results.

```
print(out,digits = 2)
```

```
## Inference for Bugs model at "/var/folders/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpyADdQp/model16243
##  2 chains, each with 10000 iterations (first 2500 discarded), n.thin = 7
##  n.sims = 2142 iterations saved
##          mu.vect sd.vect   2.5%    25%    50%    75%  97.5% Rhat n.eff
## beta[1]    -1.98    0.31  -2.55  -2.20  -2.01  -1.79  -1.31 1.00  1100
## beta[2]    -0.78    1.04  -2.69  -1.50  -0.84  -0.09   1.35 1.00  2100
## beta[3]     0.32    0.14   0.04   0.23   0.32   0.42   0.60 1.00  2100
## beta[4]    -0.75    0.18  -1.09  -0.87  -0.76  -0.64  -0.41 1.00  2100
## beta[5]     1.13    0.17   0.82   1.02   1.13   1.24   1.47 1.00  2100
## prop[1]     0.30    0.26   0.05   0.14   0.22   0.38   0.95 1.00  2100
## prop[2]     1.51    1.33   0.18   0.64   1.11   1.94   4.88 1.00  2100
## prop[3]     6.98    4.56   2.34   4.14   5.89   8.56  17.71 1.00   650
## theta[1]   -1.49    0.76  -3.05  -1.99  -1.51  -0.96  -0.05 1.00  2100
## theta[2]    0.08    0.86  -1.73  -0.44   0.10   0.66   1.58 1.00  2100
## theta[3]    1.79    0.53   0.85   1.42   1.77   2.15   2.87 1.00   560
## deviance  489.20   19.75 448.68 476.02 490.48 503.80 524.24 1.01   290
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 194.5 and DIC = 683.7
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Get posterior medians of relevant parameters.

Start w/ occupancy.

```
prop1 <- c(out$BUGSoutput$sims.array[,,'prop[1]'])
prop2 <- c(out$BUGSoutput$sims.array[,,'prop[2]'])
prop3 <- c(out$BUGSoutput$sims.array[,,'prop[3]'])
psi1 <- plogis(prop1) / (1 + plogis(prop1) + plogis(prop2) + plogis(prop3))
psi2 <- plogis(prop2) / (1 + plogis(prop1) + plogis(prop2) + plogis(prop3))
psi3 <- plogis(prop3) / (1 + plogis(prop1) + plogis(prop2) + plogis(prop3))
psi0 <- 1 - (psi1 + psi2 + psi3)
# res
res <- data.frame(post_median = c(median(psi0),
                                  median(psi1),
                                  median(psi2),
                                  median(psi3)))
rownames(res) <- c('psi00','psi10','psi01','psi11')
round(res,3)
```

```
##       post_median
## psi00       0.302
## psi10       0.170
## psi01       0.228
## psi11       0.297
```

Marginal probabilities.

```r
median(psi1 + psi3) # Pr(dolphin present)
```

```
## [1] 0.4686069
```

```r
median(psi2 + psi3) # Pr(fishing present)
```

```
## [1] 0.5258507
```

```r
median(psi2 + psi0) # Pr(dolphin absent)
```

```
## [1] 0.5313931
```

```r
median(psi1 + psi0) # Pr(fishing absent)
```

```
## [1] 0.4741493
```

Conditional probabilities.

```r
median(psi1 / (psi1 + psi0)) # Pr(dolphin present | fishing absent) ?= Pr(dolphin present)
```

```
## [1] 0.3569684
```

```r
median(psi3 / (psi3 + psi2)) # Pr(dolphin present | fishing present) ?= Pr(dolphin present)
```

```
## [1] 0.5654089
```

```r
median(psi2 / (psi2 + psi0)) # Pr(fishing present | dolphin absent) = Pr(fishing)
```

```
## [1] 0.4291839
```

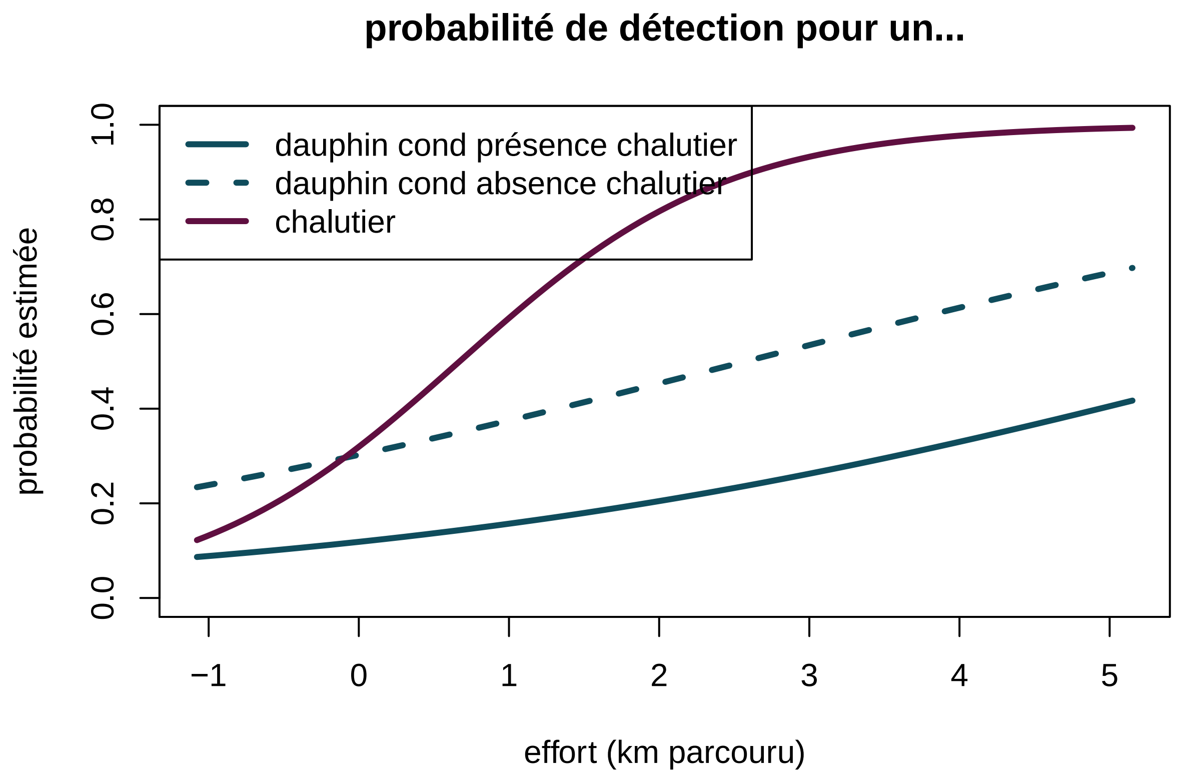
```r
median(psi3 / (psi3 + psi1)) # Pr(fishing present | dolphin present) = Pr(fishing)
```

```
## [1] 0.6387878
```

Detection.

```r
beta1 <- c(out$BUGSoutput$sims.array[,,'beta[1]'])
beta2 <- c(out$BUGSoutput$sims.array[,,'beta[2]'])
beta3 <- c(out$BUGSoutput$sims.array[,,'beta[3]'])

beta4 <- c(out$BUGSoutput$sims.array[,,'beta[4]'])
beta5 <- c(out$BUGSoutput$sims.array[,,'beta[5]'])

grid_p <- seq(range(data$eff)[1], range(data$eff)[2], length = 100)
logit_p12 <- median(beta1) + median(beta3) * grid_p
logit_p12bar <- median(beta2) + median(beta3) * grid_p
logit_p2 <- median(beta4) + median(beta5) * grid_p
```

```r
plot(grid_p, plogis(logit_p12),
     type = "l",
     col = "#0F4C5C",
     lwd = 3,
     ylim = c(0,1),
     xlab = "effort (km parcouru)",
     ylab = "probabilité estimée",
     main = "probabilité de détection pour un...")
lines(grid_p, plogis(logit_p12bar), col = "#0F4C5C", lwd = 3, lty = 2)
lines(grid_p, plogis(logit_p2), col = "#5f0f40", lwd = 3)
legend("topleft",
       col = c("#0F4C5C","#0F4C5C", "#5f0f40"),
       lty = c(1,2, 1),
       lwd = 3,
       legend = c("dauphin cond présence chalutier",
                  "dauphin cond absence chalutier",
                  "chalutier"))
```



Instead of the Waddle's formulation, I would have used $\alpha_1 + \alpha_2 z_{\text{fishing boats}}$ so that:

- B present implies $\alpha_1 + \alpha_2 = \beta_1$
- B absent implies $\alpha_1 = \beta_2$

which gives $\alpha_2 = \beta_1 - \beta_2$.

Let's have a look.

```r
alpha2 <- beta1 - beta2
mean(alpha2)
```

```
## [1] -1.206281
```

```r
median(alpha2)
```

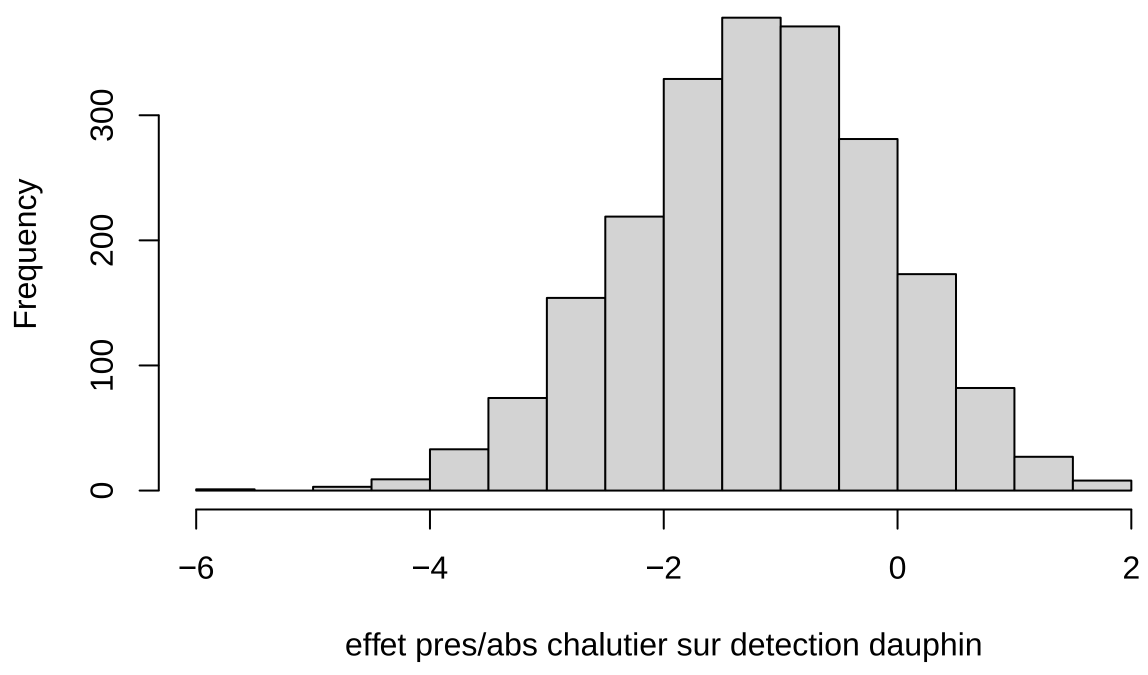
```
## [1] -1.1596
```

```r
quantile(alpha2, probs = c(2.5, 97.5) / 100)
```

```
##       2.5%      97.5%
## -3.4618388  0.8291936
```

A histogram.

```r
hist(alpha2, main = "", xlab = "effet pres/abs chalutier sur detection dauphin")
```



```r
mean(alpha2 > 0)
```

```
## [1] 0.1353875
```

Another way to formally test for the effect of the presence/absence of fishing boats on the detection of dolphins is to compare this model with a model without this effect, using the DIC or the WAIC for example.

Model w/ constant natural parameters, and detection function of sampling effort

```
model2 <- function() {

  ## state process
  for(j in 1:nsite) {
    z[j] ~ dcat(psi[1:4])
  }

  # occupancy probabilities
  psi[1] <- 1 / (1 + sum(prop[1:3])) # unoccupied
  psi[2] <- prop[1] / (1 + sum(prop[1:3])) # occupied by species A and not B
  psi[3] <- prop[2] / (1 + sum(prop[1:3])) # occupied by species B and not A
  psi[4] <- prop[3] / (1 + sum(prop[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, 1:4, z[j]])
    }
  }

  # detection matrix with obs for observations and state = true states
  # obs take values:
  # 1 for none species detected
  # 2 for species 1 detected
  # 3 for species 2 detected
  # 4 for both species detected
  # given state = unoccupied,
  for(j in 1:nsite) {
    for(k in 1:nyear) {
      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
      # given state = occupied by species A and not B,
      obs[j, k, 1, 2] <- 1 - pA[j, k] # prob obs = 1
      obs[j, k, 2, 2] <- pA[j, k] # prob obs = 2
      obs[j, k, 3, 2] <- 0 # prob obs = 3
      obs[j, k, 4, 2] <- 0 # prob obs = 4
      # given state = occupied by species B and not A,
      obs[j, k, 1, 3] <- 1 - pB[j, k] # prob obs = 1
      obs[j, k, 2, 3] <- 0 # prob obs = 2
      obs[j, k, 3, 3] <- pB[j, k] # prob obs = 3
      obs[j, k, 4, 3] <- 0 # prob obs = 4
      # given state = occupied by both species A and B,
      obs[j, k, 1, 4] <- (1 - pA[j, k]) * (1 - pB[j, k]) # prob obs = 1
      obs[j, k, 2, 4] <- pA[j, k] * (1 - pB[j, k]) # prob obs = 2
      obs[j, k, 3, 4] <- (1 - pA[j, k]) * pB[j, k] # prob obs = 3
      obs[j, k, 4, 4] <- pA[j, k] * pB[j, k] # prob obs = 4
    }
  }
  ## priors for...
  # occupancy probabilities
  for (i in 1:3){
    log(prop[i]) <- theta[i]
    theta[i] ~ dnorm(0,1)
  }
  # detection probabilities (pA function of pres/abs of B, as in Waddle et al 2010 page 1470)
  for(j in 1:nsite) {
    for(k in 1:nyear) {
      logit(pA[j, k]) <- beta[1] + beta[2] * eff[j, k]
      logit(pB[j, k]) <- beta[3] + beta[4] * eff[j, k]
    }
  }
  for (i in 1:4){
    beta[i] ~ dnorm(0,1)
  }
}
```

Initial values.

```
zinit <- apply(data$y, 1, max, na.rm = TRUE)
zinit[zinit==3] <- 4
inits <- function() {list(z = zinit,
                          beta = rnorm(4, 0, 1),
                          theta = rnorm(3, 0, 1))}
```

Parameters to be monitored.

```
params <- c("prop","theta","beta")
```

MCMC settings

```
ni <- 10000
nb <- 2500
nc <- 2
```

Run Jags from R.

```
library(R2jags)
ptm <- proc.time()
out2 <- jags(data = data,
            inits = inits,
            parameters.to.save = params,
            model.file = model2,
            n.chains = nc,
            n.iter = ni,
            n.burnin = nb)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 326
##    Unobserved stochastic nodes: 421
##    Total graph size: 8274
##
## Initializing model
```

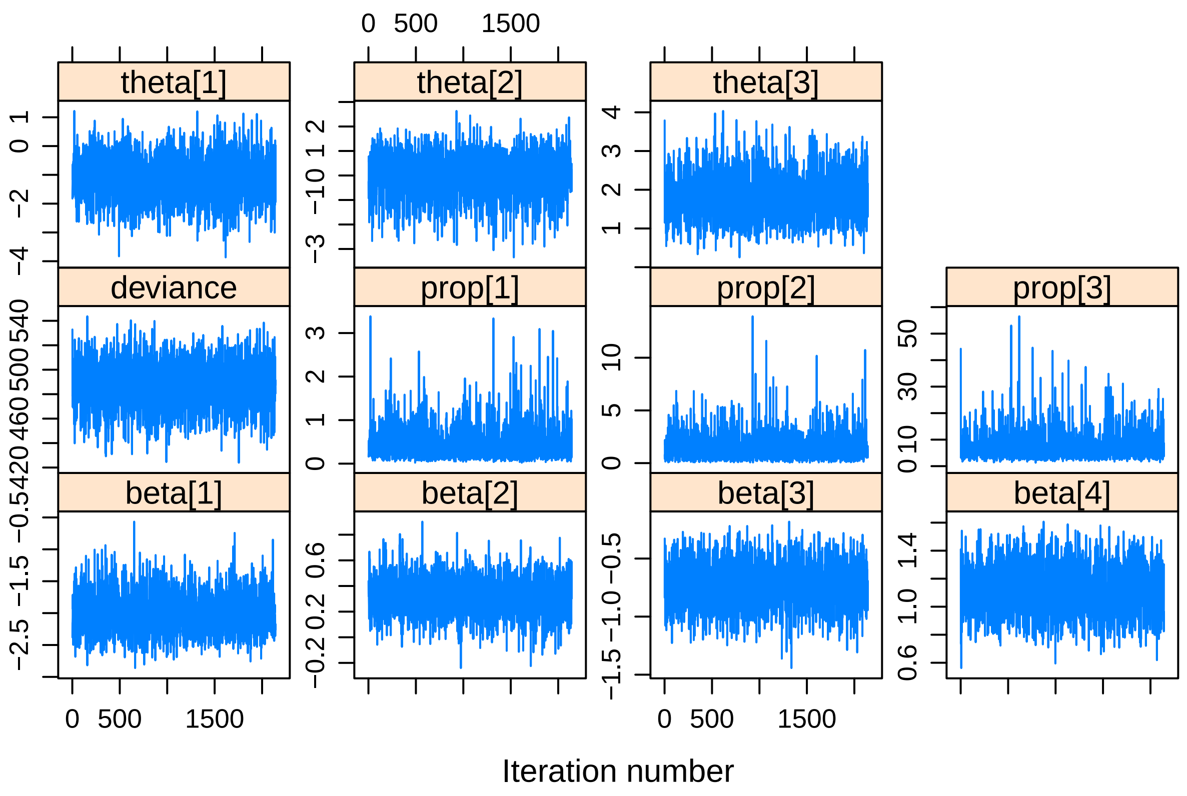
```r
x2 <- proc.time() -  ptm
x2 # 15 minutes
```

```
##    user  system elapsed
##  56.798   0.095  57.035
```

```r
save(x2, out2, file = "runwochalutier_gdegem.RData")
```

Check convergence.

```r
jagsfit.mcmc <- as.mcmc(out2$BUGSoutput$sims.matrix)
library(lattice)
xyplot(jagsfit.mcmc, layout=c(4,3))
```



Print results.

```r
print(out2,digits = 2)
```

```
## Inference for Bugs model at "/var/folders/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpyADdQp/model16243
##  2 chains, each with 10000 iterations (first 2500 discarded), n.thin = 7
##  n.sims = 2142 iterations saved
##          mu.vect sd.vect  2.5%   25%   50%   75%  97.5% Rhat n.eff
## beta[1]    -2.00    0.31 -2.54 -2.21 -2.03 -1.82  -1.35 1.01   150
## beta[2]     0.32    0.14  0.04  0.22  0.32  0.41   0.59 1.00  2100
## beta[3]    -0.73    0.18 -1.10 -0.86 -0.74 -0.62  -0.37 1.00  2100
## beta[4]     1.12    0.17  0.80  1.01  1.12  1.23   1.45 1.00  2100
## prop[1]     0.44    0.36   0.07   0.20   0.34   0.54   1.37 1.00  2100
## prop[2]     1.43    1.22   0.16   0.62   1.11   1.88   4.50 1.01   430
## prop[3]     7.11    4.94   2.29   4.13   5.84   8.46  19.81 1.01   200
## theta[1]   -1.10    0.75  -2.61  -1.63  -1.09  -0.61   0.31 1.00  2100
## theta[2]    0.03    0.86  -1.86  -0.47   0.11   0.63   1.50 1.01   430
## theta[3]    1.80    0.55   0.83   1.42   1.77   2.13   2.99 1.01   180
## deviance  487.59   18.95 449.07 474.98 488.41 500.78 522.34 1.01   170
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 178.7 and DIC = 666.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Compare the DIC values of the two models, and conclude. Needs to run the models with many more iterations.

We're almost there. The only issue we have is tha we cannot really make a nice map of the probability of co-occurrence of dolphins and fishing boats. My suggestion is to go for a model with lat/long in the model in a non-parametric relationship.

Model w/ constant natural parameters, and GAM sur lat/long; detection is function of sampling effort plus dolphin detection function of pres/abs of fishing boats

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

```r
yy_dolphin <- apply(y_dolphin, 1, max, na.rm = TRUE)
coordx <- coord[,1]
coordy <- coord[,2]
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following objects are masked from 'package:unmarked':
##
##     getData, ranef

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

```r
res <- jagam(yy_dolphin ~ s(coordx, coordy, bs = "gp"),
            family = "binomial",
            file = "psi.txt") # same structure for fishing boats and both together
#save(res, file = 'jagam.RData')
```

Specify model in BUGS language.

```
model <- function() {

  ## state process
  for(j in 1:nsite) {
    z[j] ~ dcat(psi[j, 1:4])
  }

  # occupancy probabilities
  for(j in 1:nsite) {
    psi[j, 1] <- 1 / (1 + sum(prop[j, 1:3])) # unoccupied
    psi[j, 2] <- prop[j, 1] / (1 + sum(prop[j, 1:3])) # occupied by species A and not B
    psi[j, 3] <- prop[j, 2] / (1 + sum(prop[j, 1:3])) # occupied by species B and not A
    psi[j, 4] <- prop[j, 3] / (1 + sum(prop[j, 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, 1:4, z[j]])
    }
  }

  # detection matrix with obs for observations and state = true states
  # obs take values:
  # 1 for none species detected
  # 2 for species 1 detected
  # 3 for species 2 detected
  # 4 for both species detected
  # given state = unoccupied,
  for(j in 1:nsite) {
    for(k in 1:nyear) {
      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
      # given state = occupied by species A and not B,
      obs[j, k, 1, 2] <- 1 - pA[j, k] # prob obs = 1
      obs[j, k, 2, 2] <- pA[j, k] # prob obs = 2
      obs[j, k, 3, 2] <- 0 # prob obs = 3
      obs[j, k, 4, 2] <- 0 # prob obs = 4
      # given state = occupied by species B and not A,
      obs[j, k, 1, 3] <- 1 - pB[j, k] # prob obs = 1
      obs[j, k, 2, 3] <- 0 # prob obs = 2
      obs[j, k, 3, 3] <- pB[j, k] # prob obs = 3
      obs[j, k, 4, 3] <- 0 # prob obs = 4
      # given state = occupied by both species A and B,
      obs[j, k, 1, 4] <- (1 - pA[j, k]) * (1 - pB[j, k]) # prob obs = 1
      obs[j, k, 2, 4] <- pA[j, k] * (1 - pB[j, k]) # prob obs = 2
      obs[j, k, 3, 4] <- (1 - pA[j, k]) * pB[j, k] # prob obs = 3
      obs[j, k, 4, 4] <- pA[j, k] * pB[j, k] # prob obs = 4
    }
  }
  ## 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 <- X %*% b1 ## linear predictor
  theta2 <- X %*% b2 ## linear predictor
  theta3 <- X %*% b3 ## linear predictor

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

  ## prior for s(coordx,coordy)...
  K11 <- S1[1:32,1:32] * lambda[1, 1]  + S1[1:32,33:64] * lambda[2, 1]
  K12 <- S1[1:32,1:32] * lambda[1, 2]  + S1[1:32,33:64] * lambda[2, 2]
  K13 <- S1[1:32,1:32] * lambda[1, 3]  + S1[1:32,33:64] * lambda[2, 3]
  b1[2:33] ~ dmnorm(zero[2:33], K11)
  b2[2:33] ~ dmnorm(zero[2:33], K12)
  b3[2:33] ~ dmnorm(zero[2:33], K13)
  ## smoothing parameter priors CHECK...
  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 (pA function of pres/abs of B, as in Waddle et al 2010 page 1470)
  for(j in 1:nsite) {
    B_present[j] <- equals(z[j], 3) + equals(z[j], 4)
    for(k in 1:nyear) {
      logit(pA[j, k]) <- beta[1] * B_present[j] + beta[2] * (1 - B_present[j]) + beta[3] * eff[j, k]
      logit(pB[j, k]) <- beta[4] + beta[5] * eff[j, k]
    }
  }
  for (i in 1:5){
    beta[i] ~ dnorm(0,1)
  }
}
```

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

```
# data
data <- list(y = y_jags,
             nsite = dim(y_jags)[1],
             nyear = dim(y_jags)[2],
             eff = st_effort,
             X = res$jags.data$X,
             S1 = res$jags.data$S1,
             zero = res$jags.data$zero)
```

```r
# initial values
zinit <- apply(data$y, 1, max, na.rm = TRUE)
zinit[zinit==3] <- 4
inits <- function() {list(z = zinit,
                          beta = rnorm(5, 0, 1),
                          lambda = cbind(res$jags.ini$lambda, res$jags.ini$lambda, res$jags.ini$lambda)
                          b1 = res$jags.ini$b,
                          b2 = res$jags.ini$b,
                          b3 = res$jags.ini$b)}

# parameters monitored
params <- c("beta", "b1", "b2", "b3", "lambda")

# MCMC settings
ni <- 25000
nb <- 5000
nc <- 2
```

Run Jags from R.

```r
library(R2jags)
ptm <- proc.time()
out <- jags(data = data,
            inits = inits,
            parameters.to.save = params,
            model.file = model,
            n.chains = nc,
            n.iter = ni,
            n.burnin = nb)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 326
##    Unobserved stochastic nodes: 431
##    Total graph size: 20062
##
## Initializing model
```

```r
x <- proc.time() -  ptm
x # 9 minutes
```

```
##    user  system elapsed
## 345.120   0.502 346.320
```

Save run.

```r
save(x, out, file = "runGAMgdegem.RData")
```

Check convergence:

```r
jagsfit.mcmc <- as.mcmc(out$BUGSoutput$sims.matrix)
```

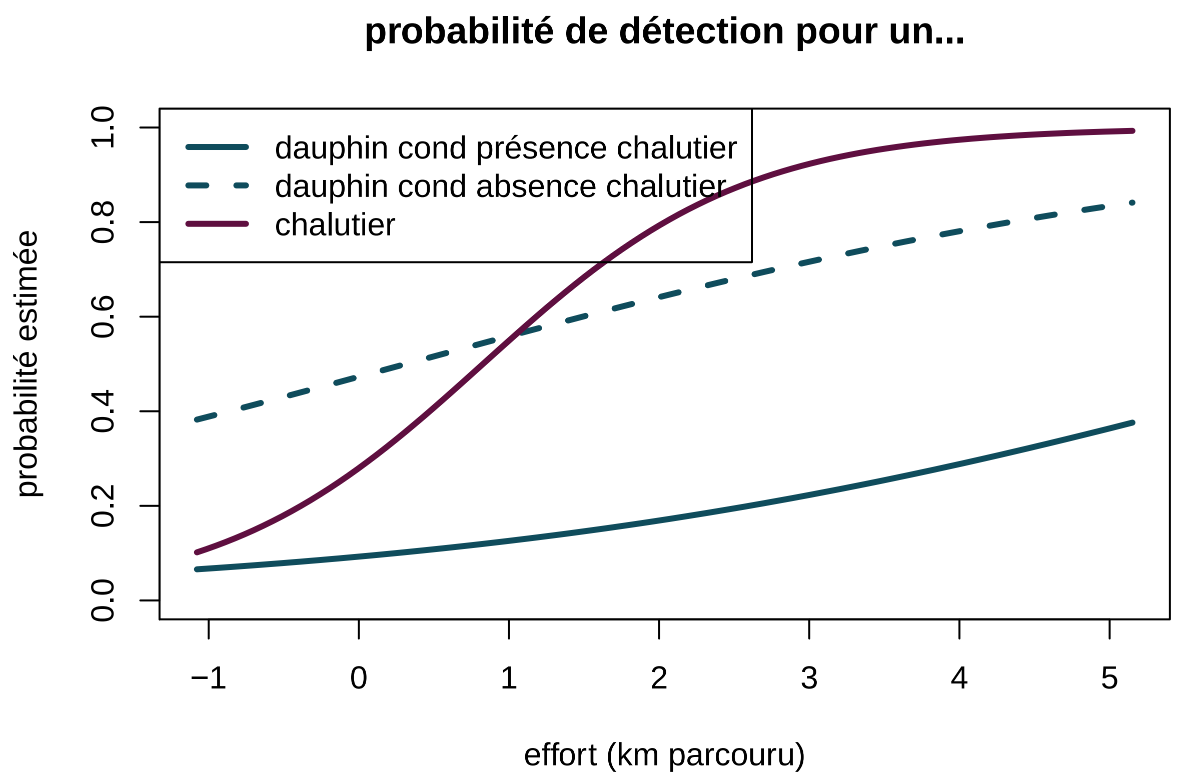
Print results.

```r
print(out,digits = 2)
```

```
## Inference for Bugs model at "/var/folders/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpyADdQp/model16243
##  2 chains, each with 25000 iterations (first 5000 discarded), n.thin = 20
##  n.sims = 2000 iterations saved
##              mu.vect sd.vect    2.5%     25%     50%     75%   97.5% Rhat n.eff
## b1[1]          -7.28    7.49  -23.57  -11.81   -6.79   -2.44    7.12 1.00  2000
## b1[2]           0.29    2.29   -4.34   -1.24    0.38    1.84    4.88 1.01  2000
## b1[3]           0.05    3.27   -6.18   -2.07   -0.09    2.17    6.58 1.01   220
## b1[4]          -0.74    7.80  -15.34   -6.12   -0.72    4.33   14.83 1.00  2000
## b1[5]          -0.33   13.29  -26.69   -8.68   -0.17    7.97   26.90 1.03    62
## b1[6]           0.52   15.33  -29.45   -8.68    0.77   10.01   32.02 1.03   140
## b1[7]           1.28   18.11  -34.48  -10.08    1.15   13.93   36.04 1.04    46
## b1[8]          10.91   31.29  -55.80   -8.51   12.33   30.72   70.37 1.44     7
## b1[9]          22.20   32.83  -43.13    1.84   23.99   43.11   86.64 1.14    19
## b1[10]          5.66   34.85  -56.39  -18.76    5.09   27.45   81.73 1.24    11
## b1[11]         16.46   34.54  -52.37   -6.22   14.30   37.87   86.67 1.08    66
## b1[12]          7.69   35.63  -71.66  -12.94   10.78   32.76   70.74 1.62     5
## b1[13]         -8.03   43.79  -94.58  -37.32   -5.72   22.22   71.64 1.54     6
## b1[14]         -9.39   51.33 -123.99  -41.66   -9.73   23.28   85.58 1.08   330
## b1[15]        -25.30   65.33 -147.67  -76.28  -20.36   26.02   91.42 2.52     3
## b1[16]        -18.56   47.01 -100.08  -51.77  -22.58   12.15   72.98 1.45     7
## b1[17]         15.69   69.37 -105.91  -31.31    9.18   54.23  180.04 1.57     6
## b1[18]          6.42   46.50  -98.66  -20.64    8.98   37.16   90.32 1.06    70
## b1[19]         29.45   80.43 -124.83  -28.36   31.64   84.69  182.72 1.75     5
## b1[20]        -26.15   77.74 -169.04  -75.42  -31.63   16.55  150.44 1.16    26
## b1[21]       -162.96   55.22 -269.65 -197.46 -164.65 -126.11  -56.86 1.08    82
## b1[22]         73.23   81.76  -74.05   19.87   70.63  121.30  259.69 1.17    13
## b1[23]        373.59   43.32  290.24  343.31  372.81  402.42  469.27 1.12    20
## b1[24]        -88.78   78.19 -220.51 -145.60  -93.73  -38.23   76.80 1.51     6
## b1[25]        -41.62  128.09 -263.86 -146.01  -44.42   53.77  196.87 3.67     2
## b1[26]       -147.74   52.95 -261.49 -177.05 -146.93 -117.17  -50.24 1.16   550
## b1[27]        -64.34   57.05 -172.97  -96.96  -68.36  -34.35   69.41 1.26    12
## b1[28]       -153.56   56.74 -271.45 -192.31 -146.91 -111.00  -56.61 1.01   180
## b1[29]        170.29   54.28   77.06  133.34  165.08  200.53  298.62 1.11    20
## b1[30]        -30.16   62.79 -178.93  -55.55  -25.07    1.26   91.83 1.26    26
## b1[31]          0.69    4.87   -8.93   -2.43    0.67    3.73    9.66 1.00   530
## b1[32]         -0.21    3.01   -8.45   -0.59   -0.01    0.45    6.55 1.03  2000
## b1[33]          0.35    3.36   -6.81   -0.38    0.02    0.65    8.93 1.00  2000
## b2[1]          -5.28    8.51  -23.11  -10.98   -4.76    0.19   11.59 1.01   190
## b2[2]           0.15    2.26   -4.37   -1.32    0.12    1.52    4.67 1.03  2000
## b2[3]          -0.38    3.22   -6.47   -2.36   -0.38    1.59    5.97 1.05  2000
## b2[4]           0.40    7.50  -13.20   -4.36    0.24    4.84   16.90 1.10    29
## b2[5]           2.67   11.66  -19.23   -5.05    2.56   10.46   27.16 1.13    17
## b2[6]           3.84   13.54  -23.10   -4.99    4.05   12.41   31.70 1.06  2000
## b2[7]           5.15   16.93  -30.91   -5.01    5.93   16.10   36.98 1.31     9
## b2[8]         -10.84   29.05  -64.76  -29.31  -12.70    6.37   53.22 1.11    32
## b2[9]           2.19   31.83  -61.90  -18.15    1.04   22.62   68.02 1.07    51
## b2[10]         -1.47   35.15  -58.65  -24.97   -6.22   18.44   81.82 1.47     7
## b2[11]         -9.84   36.62  -82.73  -32.99  -11.59   11.30   69.21 1.17    21
## b2[12]        -32.14   40.48 -115.44  -56.13  -32.12   -7.55   49.26 1.07   380
## b2[13]          3.15   47.76  -91.85  -30.93    4.34   35.70   99.01 1.73     5
## b2[14]        -24.39   58.12 -117.18  -64.37  -34.82    8.36  110.45 1.84     4
## b2[15]        -34.83   33.78  -99.99  -57.53  -36.03  -12.31   35.06 1.04    98
## b2[16]        -30.17   39.02 -105.17  -55.37  -34.80   -5.00   49.54 1.07   150
## b2[17]         -5.67  112.51 -155.55  -89.36  -46.45   82.14  228.26 3.02     3
## b2[18]         63.87   43.81  -20.16   35.46   59.08   90.78  148.78 1.84     4
## b2[19]         67.62   57.74  -44.75   28.89   64.09  104.43  183.85 2.26     3
## b2[20]        -37.57   67.50 -140.30  -92.21  -52.41   17.22   98.10 2.47     3
## b2[21]          0.29   88.61 -128.02  -58.20  -22.03   37.71  207.43 1.96     4
## b2[22]         72.01   40.68   -3.14   45.50   70.10   94.12  170.97 1.07   190
## b2[23]        254.55   84.39   91.74  194.68  256.42  315.10  409.23 2.57     3
## b2[24]       -182.41   46.41 -292.39 -210.33 -178.91 -150.29 -105.00 1.21    13
## b2[25]        -42.21  108.10 -233.43 -139.13  -15.69   53.49  115.59 4.63     2
## b2[26]       -101.81   64.80 -246.70 -141.02  -85.35  -53.78   -9.79 2.23     3
## b2[27]        -37.83   62.56 -180.78  -64.04  -28.01    0.69   65.99 1.87     4
## b2[28]        -95.57   58.39 -206.34 -135.48  -94.94  -57.07   25.18 1.14    20
## b2[29]        243.74   98.97  102.96  160.86  215.84  328.91  434.46 3.52     3
## b2[30]        -45.57   94.00 -220.47 -116.36  -46.75   23.40  138.66 3.22     3
## b2[31]         -0.35    5.24  -11.65   -3.56   -0.25    3.13    9.37 1.05  2000
## b2[32]          3.16    8.82  -11.04   -0.34    0.08    1.98   27.38 1.59     6
## b2[33]          0.73    4.27   -7.86   -0.55    0.04    1.20   12.24 1.15    16
## b3[1]           9.02    6.81    1.19    3.01    7.52   12.99   25.51 1.00  1200
## b3[2]           0.20    2.25   -4.27   -1.23    0.14    1.62    4.84 1.00  2000
## b3[3]          -0.71    3.63   -7.60   -2.99   -0.87    1.55    6.72 1.01   130
## b3[4]           2.16    8.18  -14.09   -3.28    2.40    7.92   17.25 1.01   200
## b3[5]           1.36   13.96  -25.07   -8.06    1.31   10.35   29.77 1.00   720
## b3[6]          -1.92   16.30  -35.47  -12.02   -1.40    8.80   28.52 1.16    14
## b3[7]           6.49   19.64  -32.36   -6.57    6.28   19.10   47.38 1.03    57
## b3[8]          -9.24   28.56  -68.47  -27.54  -10.30   11.88   40.18 1.51     6
## b3[9]          21.49   34.01  -42.38   -2.00   20.86   44.56   86.23 1.09    24
## b3[10]          1.13   35.10  -66.46  -21.81    0.36   21.84   76.98 1.15    19
## b3[11]         31.93   34.76  -25.61    7.87   27.58   50.91  112.39 1.42     7
## b3[12]          8.13   44.36  -75.30  -21.88    6.67   35.07  114.39 1.23    18
## b3[13]        -11.09   37.38  -74.47  -39.53  -11.14   13.73   66.10 1.07    71
## b3[14]          2.17   61.46 -119.17  -36.77    2.56   41.82  123.27 1.04    63
## b3[15]        -77.96   70.81 -232.22 -122.35  -68.96  -26.79   37.13 2.53     3
## b3[16]        -92.01   51.02 -205.15 -124.10  -83.32  -54.59  -13.93 2.27     3
## b3[17]       -117.19   58.21 -240.48 -158.38 -102.20  -74.74  -30.45 1.96     4
## b3[18]         36.81   50.07  -59.14    1.05   39.19   71.54  129.78 1.05    42
## b3[19]         69.17   55.69  -41.07   36.32   67.32   96.95  204.93 1.17  2000
## b3[20]       -156.93   44.44 -250.81 -184.69 -152.50 -127.77  -72.87 1.31     9
## b3[21]        -44.81   39.41 -113.49  -75.30  -47.46  -14.28   28.19 1.01  1200
## b3[22]         98.16   48.77   14.72   62.74   95.77  130.28  196.68 1.24    13
## b3[23]        354.04   47.04  267.45  322.26  351.99  384.22  452.59 1.03  2000
## b3[24]       -185.14   50.46 -274.76 -221.60 -189.85 -150.68  -80.59 1.07    40
## b3[25]        -30.72   64.78 -183.65  -67.08  -23.04   12.14   80.60 1.24    11
## b3[26]       -158.90   72.76 -329.30 -199.32 -138.08 -107.93  -59.91 2.07     4
## b3[27]          8.47   46.79  -84.73  -22.93    9.54   41.11   95.61 1.12    36
## b3[28]       -112.09   35.10 -185.75 -135.97 -108.48  -87.46  -50.04 1.01   440
## b3[29]        185.66   68.14   51.67  137.47  187.01  234.39  312.18 1.57     6
## b3[30]        -60.45  100.10 -244.02 -135.41  -54.29   12.67  123.92 2.27     3
## b3[31]         -0.40    5.56  -11.54   -4.03   -0.34    3.13   10.49 1.00   460
## b3[32]        -0.11    1.26   -3.59   -0.34   -0.03    0.20    3.21 1.06    33
## b3[33]        -0.29    1.27   -4.58   -0.36   -0.03    0.20    1.57 1.13    37
## beta[1]       -2.28    0.24   -2.74   -2.45   -2.28   -2.12   -1.78 1.00  2000
## beta[2]       -0.10    0.99   -1.98   -0.78   -0.11    0.56    1.90 1.00  2000
## beta[3]        0.34    0.14    0.07    0.25    0.34    0.44    0.61 1.00   490
## beta[4]       -0.93    0.19   -1.27   -1.06   -0.95   -0.83   -0.50 1.00   660
## beta[5]        1.15    0.16    0.83    1.04    1.14    1.25    1.48 1.00   600
## lambda[1,1]    0.02    0.01    0.01    0.02    0.02    0.03    0.04 1.13    16
## lambda[2,1]   28.11   70.18    0.00    0.08    1.33   15.70  270.21 1.01   140
## lambda[1,2]    0.03    0.01    0.01    0.02    0.03    0.04    0.06 2.01     4
## lambda[2,2]   22.47   85.77    0.00    0.02    0.42    8.43  189.23 1.05    49
## lambda[1,3]    0.02    0.01    0.01    0.01    0.02    0.02    0.03 1.01   470
## lambda[2,3]   27.11   56.53    0.02    0.52    4.71   26.73  194.87 1.01   230
## deviance     529.70   21.22  476.38  519.24  538.05  545.26  551.57 1.00  1600
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 225.0 and DIC = 754.7
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Get detection estimates.

```r
beta1 <- c(out$BUGSoutput$sims.array[,,'beta[1]'])
beta2 <- c(out$BUGSoutput$sims.array[,,'beta[2]'])
beta3 <- c(out$BUGSoutput$sims.array[,,'beta[3]'])

beta4 <- c(out$BUGSoutput$sims.array[,,'beta[4]'])
beta5 <- c(out$BUGSoutput$sims.array[,,'beta[5]'])

grid_p <- seq(range(data$eff)[1], range(data$eff)[2], length = 100)
logit_p12 <- median(beta1) + median(beta3) * grid_p
logit_p12bar <- median(beta2) + median(beta3) * grid_p
logit_p2 <- median(beta4) + median(beta5) * grid_p
plot(grid_p, plogis(logit_p12),
     type = "l",
     col = "#0F4C5C",
     lwd = 3,
     ylim = c(0,1),
     xlab = "effort (km parcouru)",
     ylab = "probabilité estimée",
     main = "probabilité de détection pour un...")
lines(grid_p, plogis(logit_p12bar), col = "#0F4C5C", lwd = 3, lty = 2)
lines(grid_p, plogis(logit_p2), col = "#5f0f40", lwd = 3)
legend("topleft",
       col = c("#0F4C5C","#0F4C5C", "#5f0f40"),
       lty = c(1,2, 1),
       lwd = 3,
       legend = c("dauphin cond présence chalutier",
                  "dauphin cond absence chalutier",
                  "chalutier"))
```



Let's build a nice map of the co-occurrence of dolphins and fishing boats.

First, get the whole grid.

```
grid_coord <- st_read("Grid/grid.shp") %>%
  st_transform(crs = st_crs(pays)) %>%
  st_crop(xmin = 700000, xmax = 900000, ymin = 6140000, ymax = 6300000) %>%
  st_centroid() %>%
  st_coordinates() %>%
  as_tibble() %>%
  mutate(easting = (X - meanX)/sdX,
         northing = (Y - meanY)/sdY)
```

```
## Reading layer `grid' from data source `/Users/oliviergimenez/Dropbox/Mon Mac (MacBook-Pro-de-Olivier
## Simple feature collection with 4356 features and 3 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 701000 ymin: 5886622 xmax: 1467639 ymax: 6390000
## projected CRS:  Lambert_Conformal_Conic

## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries

## Warning in st_centroid.sf(.): st_centroid assumes attributes are constant over
## geometries of x
```

Second, get linear predictor.

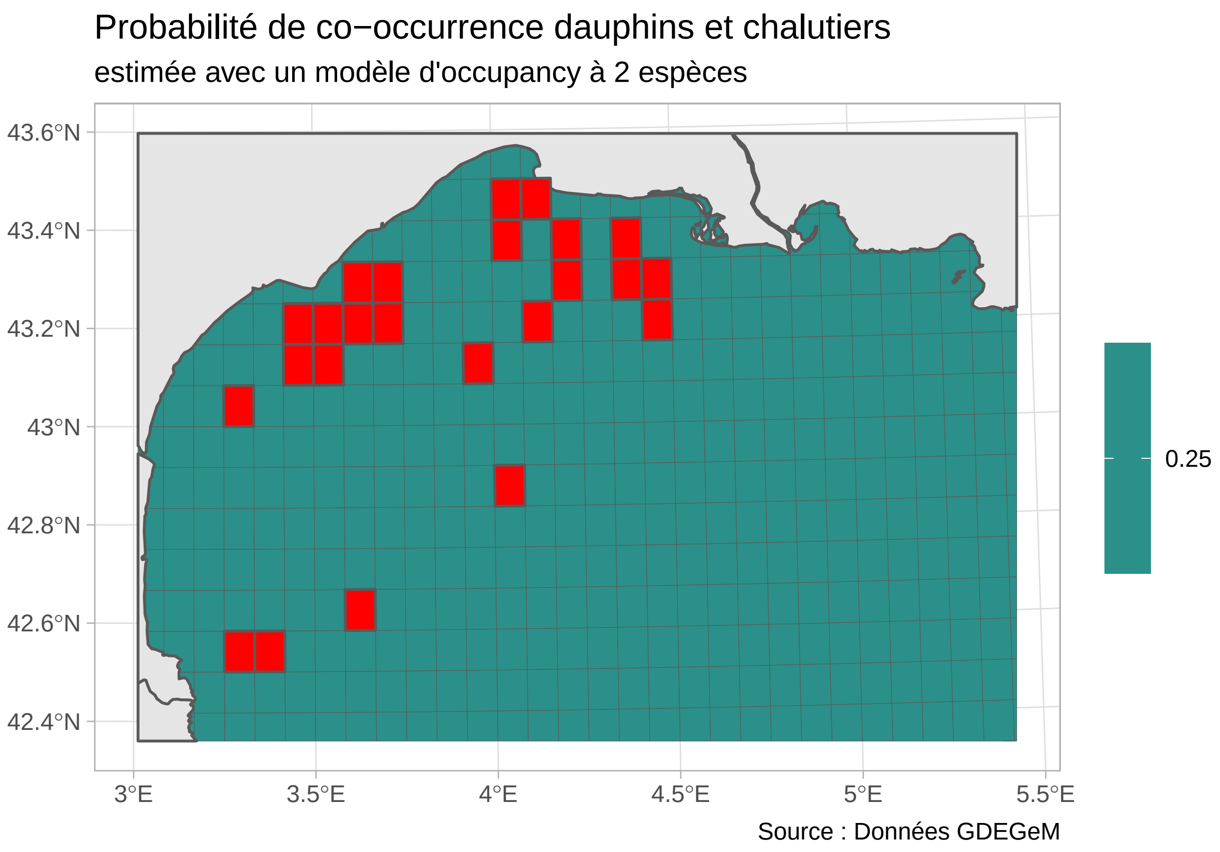
```
sm <- smoothCon(s(coordx, coordy, bs = "gp"),
                          data = data.frame(coordx = grid_coord$easting,
                                            coordy = grid_coord$northing),
                          absorb.cons = TRUE)

Xp <- PredictMat(sm[[1]], data.frame(coordx = grid_coord$easting,
                                     coordy = grid_coord$northing))
Xp <- cbind(1, Xp)
b1 <- out$BUGSoutput$sims.list$b1
b2 <- out$BUGSoutput$sims.list$b2
b3 <- out$BUGSoutput$sims.list$b3

#dim(Xp)
#dim(b1)

mu1 <- matrix(NA, nrow = nrow(Xp), ncol = nrow(b1))
mu2 <- matrix(NA, nrow = nrow(Xp), ncol = nrow(b2))
mu3 <- matrix(NA, nrow = nrow(Xp), ncol = nrow(b3))
for (i in 1:nrow(b1)){
  mu1[1:nrow(Xp), i] <- Xp %*% b1[i,]
  mu2[1:nrow(Xp), i] <- Xp %*% b2[i,]
  mu3[1:nrow(Xp), i] <- Xp %*% b3[i,]
}
```

Third, get occupancy probabilities.

```
prop1 <- apply(exp(mu1), 1, mean)
prop2 <- apply(exp(mu2), 1, mean)
prop3 <- apply(exp(mu3), 1, mean)

psi1 <- plogis(prop1) / (1 + plogis(prop1) + plogis(prop2) + plogis(prop3))
psi2 <- plogis(prop2) / (1 + plogis(prop1) + plogis(prop2) + plogis(prop3))
psi3 <- plogis(prop3) / (1 + plogis(prop1) + plogis(prop2) + plogis(prop3))
psi0 <- 1 - (psi1 + psi2 + psi3)

# Marginal probabilities.
#psi1 + psi3 # Pr(dolphin present)
#psi2 + psi3 # Pr(fishing present)
#psi2 + psi0 # Pr(dolphin absent)
#psi1 + psi0 # Pr(fishing absent)

# Conditional probabilities.
#psi1 / (psi1 + psi0) # Pr(dolphin present | fishing absent) ?= Pr(dolphin present)
#psi3 / (psi3 + psi2) # Pr(dolphin present | fishing present) ?= Pr(dolphin present)
#psi2 / (psi2 + psi0) # Pr(fishing present | dolphin absent) = Pr(fishing)
#psi3 / (psi3 + psi1) # Pr(fishing present | dolphin present) = Pr(fishing)
```

Finally, plot the map!

```
df %>%
  ggplot() +
  geom_sf(data = grid, lwd = 0.1, aes(fill = psi3)) +
  geom_sf(data = pays) +
  scale_fill_viridis_c(name = "") +
  geom_sf(data = grid %>% slice(cooc), fill = "red") +
  labs(title = "Probabilité de co-occurrence dauphins et chalutiers",
       subtitle = "estimée avec un modèle d'occupancy à 2 espèces",
       caption = "Source : Données GDEGeM")
```


Probabilité de co−occurrence dauphins et chalutiers
estimée avec un modèle d'occupancy à 2 espèces
43.6°N
43.4°N
43.2°N
43°N
42.8°N
42.6°N
42.4°N
3°E
3.5°E
4°E
4.5°E
5°E
5.5°E
0.25
Source : Données GDEGeM