Sensitivity of spatial capture-recapture models to errors in trap activity

Olivier Gimenez

10/12/2020

Introduction

In France and Switzerland, we have camera trap surveys that are conducted to estimate lynx density. For intensive/deterministic surveys, the period of activity of traps is known. For opportunistic surveys, we most often do not know the period of activity of traps.

Our objective is to combine data from both surveys, intensive/deterministic and opportunistic surveys.

Before analyzing real data, we will conduct a simulation study to assess the sensitivity of spatial capture-recapture models to errors in the determination of the duration of activity of camera traps.

In other words, we would like to estimate the bias in density estimates when we falsely assume that all traps are active all the time, while in reality some of them were inactive for some time due to malfunctions or rotations.

Below I provide some preliminary results that suggest spatial capture-recapture models are robust to over-estimation of trap activity duration. Check out the figure at the bottom of the document.

However, further work should be done before we can safely do the combination. In particular:

- Can we build profiles of camera-trap users for opportunistic sampling that would allow us predicting trap activity with respect to some explanatory variables (age, profession, season, motivation, etc).
- Can we think of scenarios to mimic interruption in trap activity (clustering in space, in habitat, etc).
- Increase number of simulations.

Build functions

Load the `oSCR` package that will be used to fit spatial capture-recapture models.

```
#devtools::install_github("https://github.com/jaroyle/oSCR")
library(oSCR)
```

```
##
## Attaching package: 'oSCR'
```

```
## The following object is masked from 'package:purrr':
##
##     flatten
```

We first build a function `simul.scr` that simulates data from a spatial capture-recapture experiment. Besides standard parameters like population size, baseline detection probability, spatial scale, the number of capture occasions and the size of the state space, we also give control upon two important parameters:

- `inactive.from.time` defines the occasions from which traps become inactive, and
- `percent.inactive.traps` defines the percentage of inactive traps.

Note that the netword of traps is built by substracting a buffer to the state-space via `expand.grid(x = 3:(upper.x - 3), y = 3:(upper.y - 3))` where `upper` gives the upper bound of the state-space along the corresponding dimension.

For completeness, we simulate a dataset with the correct information on trap activity, and another one with the incorrect information on trap activity, which is basically all traps are active all the time.

The function provides all the ingredients required by package `oSCR` to fit models.

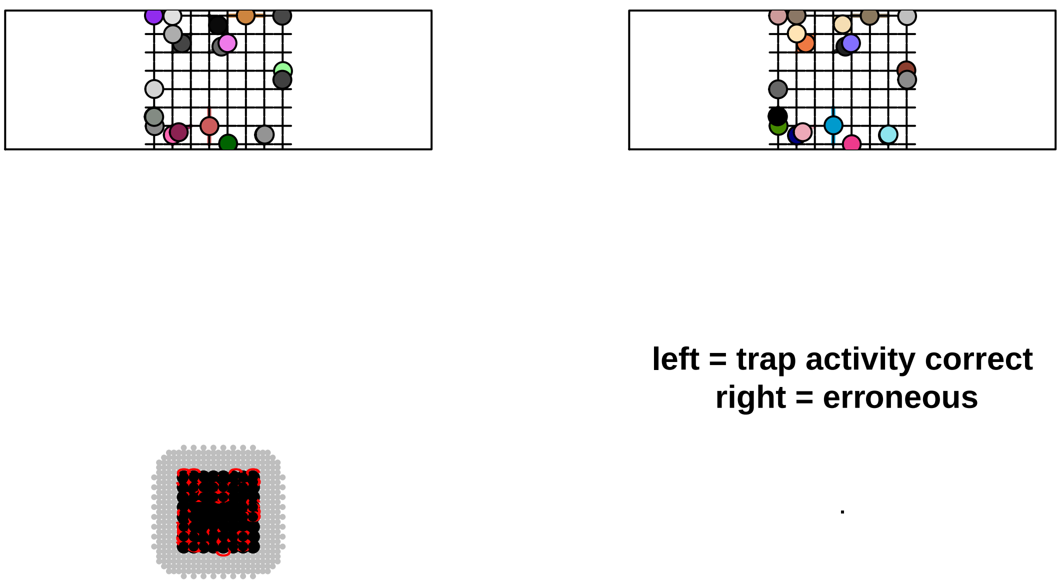
```
simul.scr <- function(pop.size = 40, # pop size
                      baseline.detection = 0.2, # baseline detection
                      spatial.scale = 0.6, # spatial scale parameter
                      nb.occ = 10, # no.occasions
                      inactive.from.time = 5,
                      percent.inactive.traps = 50,
                      upper.x = 13, # upper bound of the state-space along x-axis
                      upper.y = 13, # upper bound of the state-space along y-axis
                      make.plot = FALSE){
  xlim <- c(0, upper.x)
  ylim <- c(0, upper.y) # state space
  N <- rpois(1, pop.size) # population size
  Dens <- pop.size / (upper.x * upper.y) # density
  s <- data.frame(x = runif(N, xlim[1], xlim[2]),
                  y = runif(N, ylim[1], ylim[2])) # activity centers
  traps <- expand.grid(x = 3:(upper.x - 3),
                       y = 3:(upper.y - 3)) # trap locations
  D <- e2dist(s, traps) # distance to traps
  pmat <- baseline.detection * exp(-D * D / (2 * spatial.scale * spatial.scale)) # prob(capture in trap
  J <- nrow(traps) # no. traps
  y <- array(0, dim = c(N, J, nb.occ)) # empty enc array
  trapactiv <- matrix(1, nrow = J, ncol = nb.occ) # all traps active all the time
  mask <- sample(J, round(J * percent.inactive.traps / 100))
  if (length(mask) == 0)  {
    trapactiv2 <- trapactiv}
  else  {
    trapactiv2 <- trapactiv
    trapactiv2[mask, inactive.from.time:nb.occ] <- 0 # half the traps get inactive half way
  }
  for(i in 1:N){
    for(j in 1:J){
      for (k in 1:nb.occ){
        y[i,j,k]<- rbinom(1, 1, pmat[i,j] * trapactiv2[j,k]) # simulate [i,j,k] encounters
      }
    }
  }
  captured <- apply(y, 1, sum) > 0 # identify captured inds
  y <- y[captured,,] # removed uncaptured inds
```

```r
edf <- data.frame(which(y > 0, arr.ind = T)) # which cells are non-0 (caps)
edf <- data.frame(session = rep(1, nrow(edf)), # add the session column
                  ind = edf$dim1, # row names are individual IDs
                  trap = edf$dim2, # col names are trap IDs
                  occ = edf$dim3) #slice names are occasion IDs
tdf.true <- data.frame(Trap_id = 1:nrow(traps),
                       X = traps[,1],
                       Y = traps[,2],
                       trapOperation = trapactiv2)
tdf.false <- data.frame(Trap_id = 1:nrow(traps),
                        X = traps[,1],
                        Y = traps[,2],
                        trapOperation = trapactiv)
# format simulated data
data.true <- data2oscr(edf = edf,
                       tdf = list(tdf.true),
                       sess.col = 1,
                       id.col = 2,
                       occ.col = 4,
                       trap.col = 3,
                       K = nb.occ,
                       ntraps = nrow(tdf.true))
sf.true <- data.true$scrFrame
data.false <- data2oscr(edf = edf,
                        tdf = list(tdf.false),
                        sess.col = 1,
                        id.col = 2,
                        occ.col = 4,
                        trap.col = 3,
                        K = nb.occ,
                        ntraps = nrow(tdf.false))
sf.false <- data.false$scrFrame
ss.buffer.true <- make.ssDF(scrFrame = sf.true, # the scrFrame created above
                            buffer = 3, # 3.00 km (sigma = 0.6km)
                            res = 0.5) # 0.5 km
ss.buffer.false <- make.ssDF(scrFrame = sf.false, # the scrFrame created above
                            buffer = 3, # 3.00 km (sigma = 0.6km)
                            res = 0.5) # 0.5 km
if (make.plot == TRUE){
  par(mfrow = c(2,2))
  # Encounters
  plot(sf.true, ax = FALSE, jit = 1)
  plot(sf.false, ax = FALSE, jit = 1)
  # State-space, traps and captures
  plot(ss.buffer.true, sf.true, spider = TRUE)
  plot(1, 1, pch = '.',
       axes = FALSE,
       xlab = "",
       ylab= "",
       main = "left = trap activity correct\n right = erroneous")
  #plot(ss.buffer.false, sf.false, spider = TRUE)
  }
list(sf.true = sf.true,
```

```r
        sf.false = sf.false,
        ss.buffer.true = ss.buffer.true,
        ss.buffer.false = ss.buffer.false,
        density = Dens,
        pop.size = pop.size)
}
```

Let's illustrate the use of our function.

```r
set.seed(1234)
sim <- simul.scr(make.plot = TRUE)
```




```r
str(sim, max.level = 1)
```

```
## List of 6
##  $ sf.true        :List of 11
##   ..- attr(*, "class")= chr "scrFrame"
##  $ sf.false       :List of 11
##   ..- attr(*, "class")= chr "scrFrame"
##  $ ss.buffer.true :List of 1
##   ..- attr(*, "class")= chr "ssDF"
##  $ ss.buffer.false:List of 1
##   ..- attr(*, "class")= chr "ssDF"
##  $ density        : num 0.237
##  $ pop.size       : num 40
```

Now we build another function `compute.bias` that computes the relative bias (in percentage) in the parameters of a spatial capture-recapture model with constant parameters: population size, density, baseline detection and scale.

We simulate data, fit both models with correct and incorrect information on trap activity, and compute bias. This process is repeated `nb.simul` times.

```r
compute.bias <- function(nb.simul = 2,
                         pop.size = 40,
                         baseline.detection = 0.2, # baseline detection
                         spatial.scale = 0.6, # spatial scale parameter
                         nb.occ = 10, # no.occasions
                         inactive.from.time = 5,
                         percent.inactive.traps = 50,
                         upper.x = 13,
                         upper.y = 13){

  # pre-allocate memory
  res.true <- matrix(NA, nrow = nb.simul, ncol = 4)
  colnames(res.true) <- c("density", "abundance", "detection", "scale")
  res.false <- res.true

  for (i in 1:nb.simul){

    # simulate data
    sim <- simul.scr(pop.size = 40,
                     baseline.detection = 0.2, # baseline detection
                     spatial.scale = 0.6, # spatial scale parameter
                     nb.occ = 10, # no.occasions
                     inactive.from.time = 5,
                     percent.inactive.traps = 50,
                     upper.x = 13,
                     upper.y = 13,
                     make.plot = FALSE)

    # fit model with correct info on trap activity, and compute bias
    scr0 <- oSCR.fit(list(D ~ 1, p0 ~ 1, sig ~ 1), scrFrame = sim$sf.true, ssDF = sim$ss.buffer.true)
    # get estimated density
    pred.df.dens <- data.frame(Session = factor(1))
    pred.dens <- get.real(scr0, type = "dens", newdata = pred.df.dens, d.factor = 4)
    # get estimated abundance
    pred.n <- get.real(scr0, type = "dens", newdata = pred.df.dens, d.factor = nrow(scr0$ssDF[[1]]))
    # get estimated encounter probability at d(x,s) = 0
    pred.df.det <- data.frame(Session = factor(1))
    pred.det <- get.real(scr0, type = "det", newdata = pred.df.det)
    # get estimated spatial scale parameter
    pred.df.sig <- data.frame(Session = factor(1))
    pred.sig <- get.real(scr0, type = "sig", newdata = pred.df.sig)
    res.true[i, 1] <- unlist(pred.dens[1])
    res.true[i, 2] <- unlist(pred.n[1])
    res.true[i, 3] <- unlist(pred.det[2])
    res.true[i, 4] <- unlist(pred.sig[1])

    # fit model with erroneous info on trap activity, and compute bias
    scr0 <- oSCR.fit(list(D ~ 1, p0 ~ 1, sig ~ 1), scrFrame = sim$sf.false, ssDF = sim$ss.buffer.false)
    pred.df.dens <- data.frame(Session = factor(1))
    pred.dens <- get.real(scr0, type = "dens", newdata = pred.df.dens, d.factor = 4)
    pred.n <- get.real(scr0, type = "dens", newdata = pred.df.dens, d.factor = nrow(scr0$ssDF[[1]]))
    pred.df.det <- data.frame(Session = factor(1))
    pred.det <- get.real(scr0, type = "det", newdata = pred.df.det)
```

```
    pred.df.sig <- data.frame(Session = factor(1))
    pred.sig <- get.real(scr0, type = "sig", newdata = pred.df.sig)
    res.false[i, 1] <- unlist(pred.dens[1])
    res.false[i, 2] <- unlist(pred.n[1])
    res.false[i, 3] <- unlist(pred.det[2])
    res.false[i, 4] <- unlist(pred.sig[1])
  }
  list(res.true = res.true,
       res.false = res.false,
       bias.true = (apply(res.true, 2, mean) - c(sim$density, sim$pop.size, baseline.detection, spatial
       bias.false = (apply(res.false, 2, mean) - c(sim$density, sim$pop.size, baseline.detection, spatia
}
```

Simulation study

Let's define some scenarios. We assume that our experiment runs over 10 capture occasions. We consider all combinations with percent.inactive.traps = 30%, 40%, 50%, 60%, 70%, 80% and inactive.from.time = 5, 6, 7, 8.

```
inactive.from.time <- 5:8
percent.inactive.traps <- seq(30, 80, by = 10)
grid <- expand.grid(inactive.from.time, percent.inactive.traps)
colnames(grid) <- c("inactive.from.time", "percent.inactive.traps")
```

We run our experiment with 50 simulations. It lasted around 17 hours on my computer.

```
sim.res <- list()
start_time <- Sys.time()
for (i in 1:nrow(grid)){
  sim.res[[i]] <- compute.bias(nb.simul = 50,
                      inactive.from.time = grid[i, 1],
                      percent.inactive.traps = grid[i, 2])
}
end_time <- Sys.time()
duration <- end_time - start_time
save(sim.res, duration, file = "sim.RData")
```

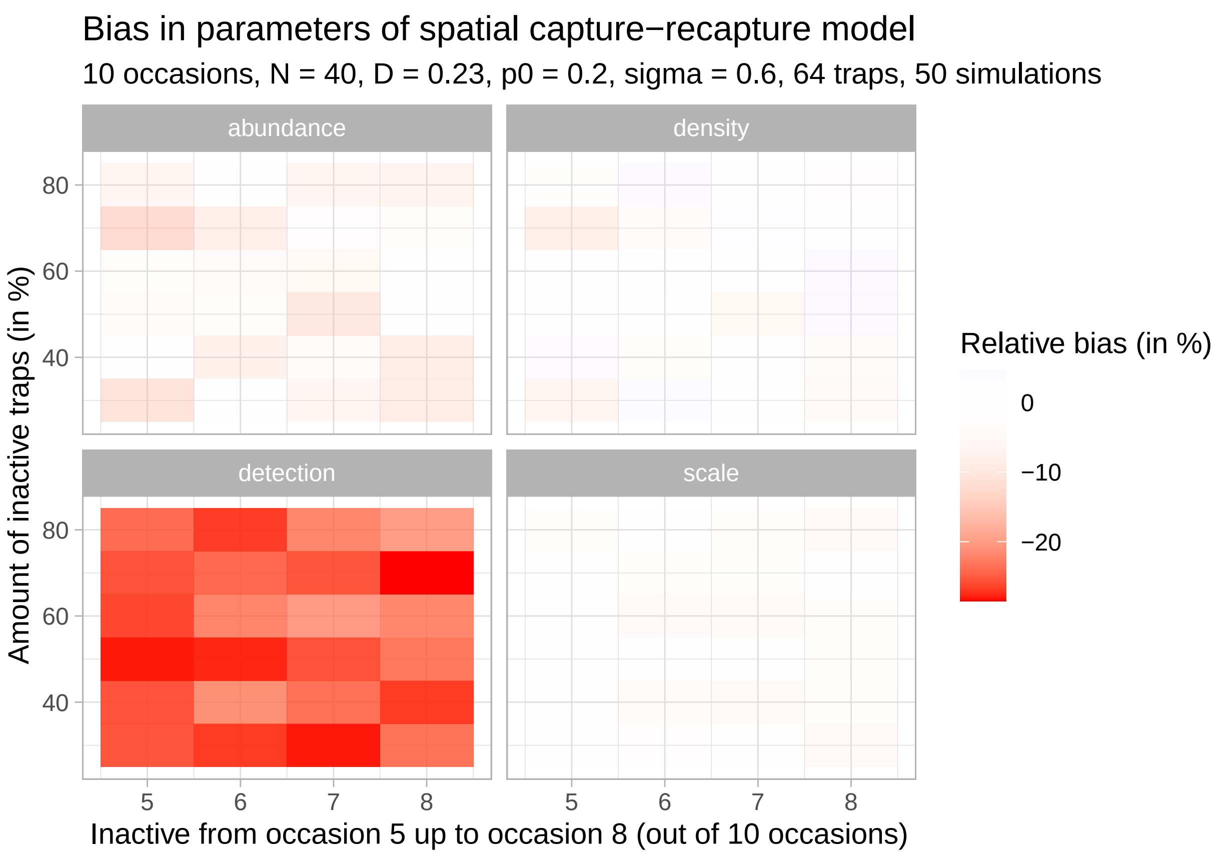
We format the results for visualisation.

```
load("sim.RData")
biastrue <- NULL
biasfalse <- NULL
for (i in 1:length(sim.res)){
  biastrue <- rbind(biastrue, sim.res[[i]]$bias.true)
  biasfalse <- rbind(biasfalse, sim.res[[i]]$bias.false)
}
biastrue <- data.frame(biastrue,
                  model = rep("correct",nrow(biastrue)),
                  grid)
biasfalse <- data.frame(biasfalse,
                  model = rep("incorrect",nrow(biasfalse)),
                  grid)
```

```r
df <- bind_rows(biastrue, biasfalse)
df <- df %>% pivot_longer(cols = density:scale,
                         names_to = "par",
                         values_to = "bias")
```

And tada!

```r
df %>%
  filter(model == "incorrect") %>%
  ggplot() +
  aes(x = inactive.from.time,
      y = percent.inactive.traps,
      fill = bias) +
  geom_tile() +
  labs(y = "Amount of inactive traps (in %)",
       x = "Inactive from occasion 5 up to occasion 8 (out of 10 occasions)",
       fill = "Relative bias (in %)",
       title = "Bias in parameters of spatial capture-recapture model",
       subtitle = "10 occasions, N = 40, D = 0.23, p0 = 0.2, sigma = 0.6, 64 traps, 50 simulations") +
  scale_fill_gradient2(low = "red",
                       mid = 0,
                       high = "blue") +
  facet_wrap(vars(par), nrow = 2)
```



Baseline detection is highly negatively biased, that is, it is estimated much lower than it should be. It is what we expect because the 0s that we have while the traps are inactive are taken as non-detections by the model when we consider that all traps are active all the time.

It sounds like our parameter of prime interest, density, is little affected whatever the scenario considered.