Practical presence-only data

Biodiversity modelling

F.G. Blanchet – August 19–23, 2019

Introduction

This practical document presents R code for the section of the course regarding presence-only data.

Different examples will be given using the course data for each of the approach discuss during the course.

In the present document, we will focus on the distribution of *Passerella iliaca*.

Load R packages

```
library(sp)
library(raster)
```

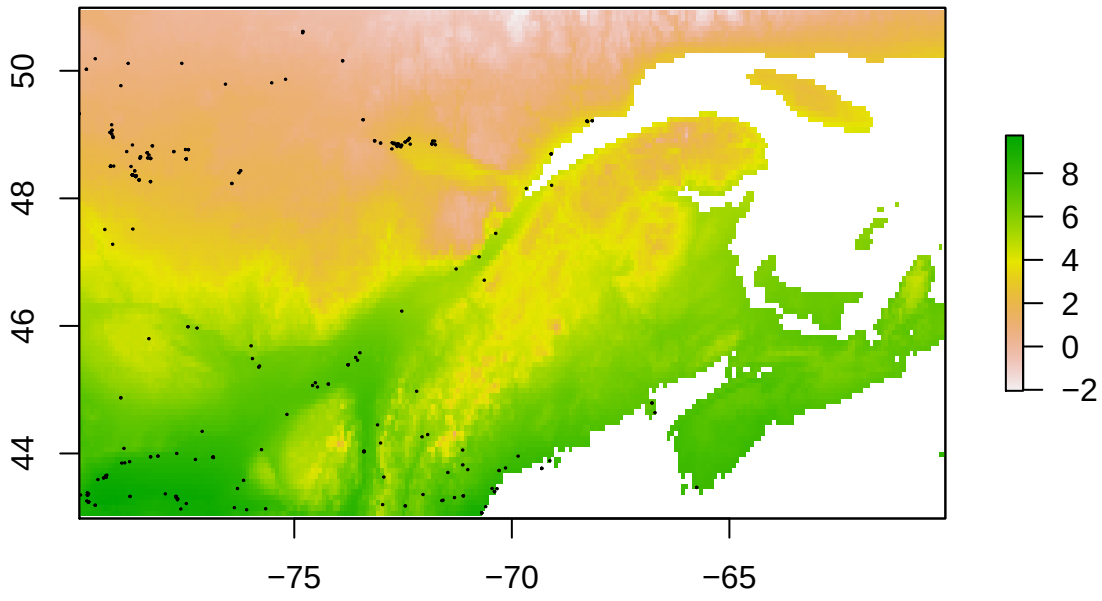
Load the data

```
bird <- readRDS("bird5.RDS")
climatePresent <- readRDS("climate_Present.RDS")
road <- readRDS("road_Distance.RDS")
```

Let's look at data we have for *Passerella iliaca*

```
sp <- bird$Oporornis.agilis

plot(climatePresent[[1]])
points(sp, pch = 19, cex = 0.1)
```



Survey region

For point process models, it is important to gather some information about the survey region. Specifically, knowing the area of the survey area is important to know

```
library(maptools)
```

```
## Checking rgeos availability: TRUE
```

```
# Construct a SpatialPolygons object
landPoly <- rasterToPolygons(climatePresent[[1]])
landPoly <- unionSpatialPolygons(landPoly,
                                 ID = rep(1,length(landPoly)))

# Find the area of the survey region
areaRegion <- raster::area(landPoly)/1000000
```

Focus only on the occurences that fall into the survey regions

```
spRegion <- intersect(sp, landPoly)
```

Define quadrature points

```
# Define random samples
set.seed(12)
xSmpl <- runif(20000, xmin(landPoly), xmax(landPoly))
```

```
ySmpl <- runif(20000, ymin(landPoly), ymax(landPoly))

# Organise them into a SpatialPoints object
xySmpl <- SpatialPoints(cbind(xSmpl, ySmpl),
                        proj4string = landPoly@proj4string)

# Find which one is in landPoly
quadIntersect <- intersect(xySmpl, landPoly)

# Select 10000 of the selected points
nquad <- 10000

quadSel <- coordinates(quadIntersect)[1:nquad,]
```

Extracting the climate data for the region of interest

This involves extracting the data for the presence points but also for the quadrature points.

```
# Organise coordinates
spQuadxy <- rbind(coordinates(spRegion), quadSel)

# Extract climate variables
climateSpQuad <- extract(scale(climatePresent),
                         spQuadxy,
                         method = "simple")

# Organize climate data
climateSpQuad <- as.data.frame(climateSpQuad)
```

Using weighted GLMs

Downweighted Poisson regression

```
# Build response variable
spQuad <- c(rep(1, length(spRegion)), rep(0, nquad))

# Build weight
weight <- rep(1/nquad, length(spQuad))
weight[spQuad == 0] <- areaRegion / nquad

# Point process model
sel <- sample(length(spQuad), 1000)

spModel <- glm(spQuad/weight ~.,
```

```
                data = climateSpQuad,
                family = poisson(),
                weights = weight)
```

Are there enough quadrature points?

```
# Climate data at quadrature points
climateQuad <- climateSpQuad[-(1:length(spRegion)),]

# Calculate the estimated intensity at the quadrature points
intensityQuad <- predict(spModel,
                         newdata = climateQuad,
                         type = "response")

# Standard deviation of the estimated intensity at the quadrature points
sdIntensityQuad <- sd(intensityQuad)

nquadWanted <- areaRegion^2 * sdIntensityQuad^2 / qnorm(0.975)
nquadWanted
```

```
## [1] 185001.3
```

Redefine quadrature points

```
# Define random samples
set.seed(12)
xSmpl <- runif(300000, xmin(landPoly), xmax(landPoly))
ySmpl <- runif(300000, ymin(landPoly), ymax(landPoly))

# Organise them into a SpatialPoints object
xySmpl <- SpatialPoints(cbind(xSmpl, ySmpl),
                        proj4string = landPoly@proj4string)

# Find which one is in landPoly
quadIntersect <- intersect(xySmpl, landPoly)

# Select 200000 of the selected points
nquad <- 200000

quadSel <- coordinates(quadIntersect)[1:nquad,]
```

Extracting the climate data for the region of interest

This involves extracting the data for the presence points but also for the quadrature points.

```r
# Organise coordinates
spQuadxy <- rbind(coordinates(spRegion), quadSel)

# Extract climate variables
climateSpQuad <- extract(scale(climatePresent),
                         spQuadxy,
                         method = "simple")

# Organize climate data
climateSpQuad <- as.data.frame(climateSpQuad)
```

Recalculate the downweighted Poisson regression

```r
# Build response variable
spQuad <- c(rep(1, length(spRegion)), rep(0, nquad))

# Build weight
weight <- rep(1/nquad, length(spQuad))
weight[spQuad == 0] <- areaRegion / nquad

# Point process model
sel <- sample(length(spQuad), 1000)

spModel <- glm(spQuad/weight ~.,
               data = climateSpQuad,
               family = poisson(),
               weights = weight)
```

Check if enough quadrature points were used

```r
# Climate data at quadrature points
climateQuad <- climateSpQuad[-(1:length(spRegion)),]

# Calculate the estimated intensity at the quadrature points
intensityQuad <- predict(spModel,
                         newdata = climateQuad,
                         type = "response")

# Standard deviation of the estimated intensity at the quadrature points
sdIntensityQuad <- sd(intensityQuad)

SE <- areaRegion * sdIntensityQuad / sqrt(nquad)
SE
```

```
## [1] 1.36245
```

Perfect !

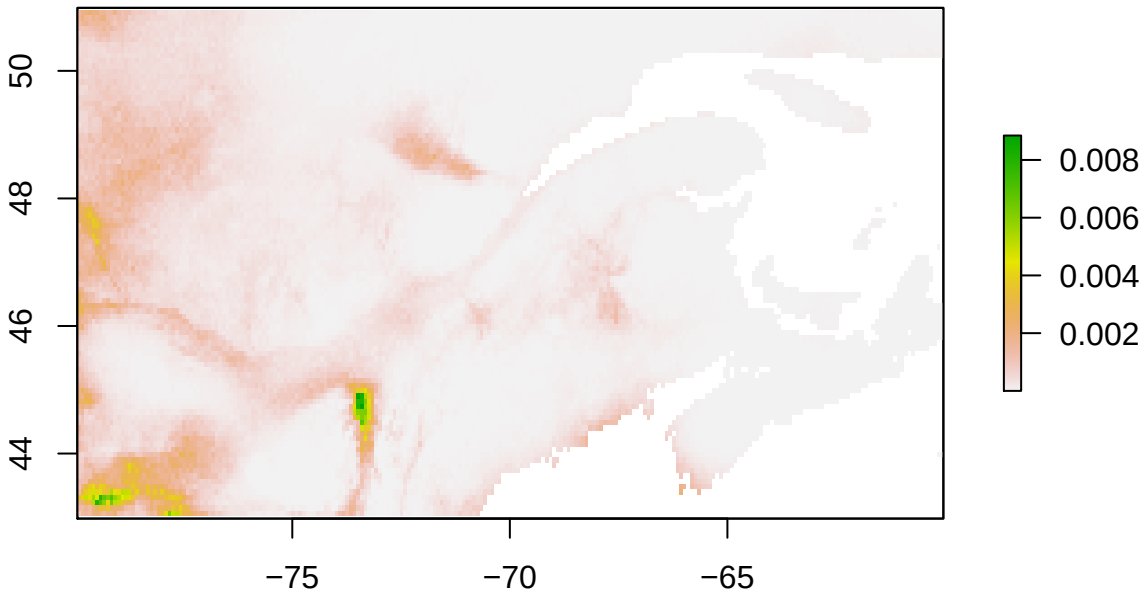
Draw a map of the model

```
climateDat <- as.data.frame(scale(values(climatePresent)))

# Calculate the estimated intensity for the survey region
intensityMap <- predict(spModel,
                        newdata = climateDat,
                        type = "response")

# Build the raster object
pred <- raster(climatePresent)
values(pred) <- intensityMap

# Draw the map
plot(pred)
```



Because this model assumes all occurrence points are independent, let's take this into account.

Warning

The code in the following section is very (!) long to run. It was included for you to know how to do these types of models in R, I recommend not to run it during the course.

Area-interaction model

To estimate an area-interaction model we need to use the `spatstat` R package.

```
library(spatstat)
```

To fit an area-interaction model, we first need to define the best radius to use. For illustration purposes, let's assume a radius of 10 km

```
# Build owin object
landWin <-as.owin(landPoly)

# Define point process object
spPPP <- ppp(coordinates(sp)[,1],
             coordinates(sp)[,2],
             window = landWin,
             check = FALSE)

# Define quadrature point object
quadPoint <- ppp(quadSel[,1], quadSel[,2],window = landWin)
Q <- quadscheme(data = spPPP, dummy = quadPoint, method = "grid")

# Formula
formu <- as.formula(paste0("~", paste0("bio",1:19, collapse = "+")))

# Format climate
climateList <- vector("list", length = nlayers(climatePresent))
for(i in 1:nlayers(climatePresent)){
  climateList[[i]] <- as.im(climatePresent[[i]])
}

names(climateList) <- paste0("bio", 1:19)

AImodel <- ppm(Q,
               trend = formu,
               covariates = climateList,
               interaction = AreaInter(0.1))
```