Practical - Joint species distribution models

Biodiversity modelling

F.G. Blanchet – August 19–23, 2019

Introduction

This practical document presents R code for the section of the course regarding models for presence-absence data using multiple species.

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

Load R packages

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

Load the data

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

Extract and organize the data

Let's organize the data based on what we learned from logistic regression.

```
# Pixels within 10 km of roads
roadDat <- values(road)
road10 <- which(roadDat < 10000)
roadSub <- raster(road)
values(roadSub)[road10] <- values(road)[road10]
roadSub <- mask(roadSub, climatePresent[[1]])
locPixRoad <- which(!is.na(values(roadSub)))

# Species data
spDat <- values(bird)
spSub <- spDat[locPixRoad,]

# Climate data
```

```r
climateAll <- values(climatePresent)
climateDat <- scale(climateAll)
climateSub <- climateDat[locPixRoad,]
colnames(climateSub) <- colnames(climateDat)
climateSub <- as.data.frame(climateSub)

# Reduce the number of climate variables using a PCA
library(vegan)
```

```
## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.5-5
```

```r
NALoc <- which(is.na(climateDat[,1]))
climatePCA <- rda(climateDat[-NALoc,], na.rm = TRUE)
cumsum(eigenvals(climatePCA))/sum(eigenvals(climatePCA))
```

```
##       PC1       PC2       PC3       PC4       PC5       PC6       PC7
## 0.5506398 0.8074941 0.8982391 0.9410703 0.9742095 0.9900023 0.9944899
##       PC8       PC9      PC10      PC11      PC12      PC13      PC14
## 0.9960518 0.9973619 0.9986228 0.9992817 0.9995830 0.9997494 0.9998775
##      PC15      PC16      PC17      PC18
## 0.9999534 0.9999844 0.9999981 1.0000000
```

```r
# The first three PCA axes should do the trick
climateAxes <- scores(climatePCA,
                      choices = 1:3,
                      display = "sites")

# Reorganise the climate data
climateAxesAll <- matrix(NA, nrow = nrow(climateDat), ncol = 3)
climateAxesAll[-NALoc,] <- climateAxes
climateAxesRoad <- climateAxesAll[locPixRoad,]
climateAxesSub <- climateAxesRoad[-NALoc,]
colnames(climateAxesSub) <- paste0("PC",1:3)

# Remove NAs from species data
spNoNA <- spSub[-NALoc,]
```

Build the model

To perform hierarchical modelling of species community you need to load the R package `HMSC`.

```r
library(HMSC)
```

HMSC using only environmental variables

The first thing to do is to format the data (response and explanatory variables) for it to be used by HMSC.

Format the data for HMSC

```r
# Format data for HMSC
set.seed(12)
formDat <- as.HMSCdata(Y = spNoNA, X = climateAxesSub)
```

```
## [1] "row names were added to 'Y'"
## [1] "row names were added to 'X'"
```

If you are OK with the weakly informative prior specification, we can run the model. If not, you can specify the priors

```r
priorSpec <- as.HMSCprior(formDat,
                          family = "probit",
                          meansParamX = matrix(rep(0, ncol(formDat$X))))
```

Note that the link function used for presence-absence data is not the logit link but the probit link. The difference between the logit and probit link is minimial. It was decided to implement the probit instead of the logit model because it is simpler to implement within an MCMC framework.

Run the model

To run the model, we need to decide on how many iterations we should perform, including burn-in and thinning.

```r
spClimate <- hmsc(formDat,
                  priors = priorSpec,
                  family = "probit",
                  niter = 2000,
                  nburn = 1000,
                  verbose = FALSE)
```

Check for parameter convergence

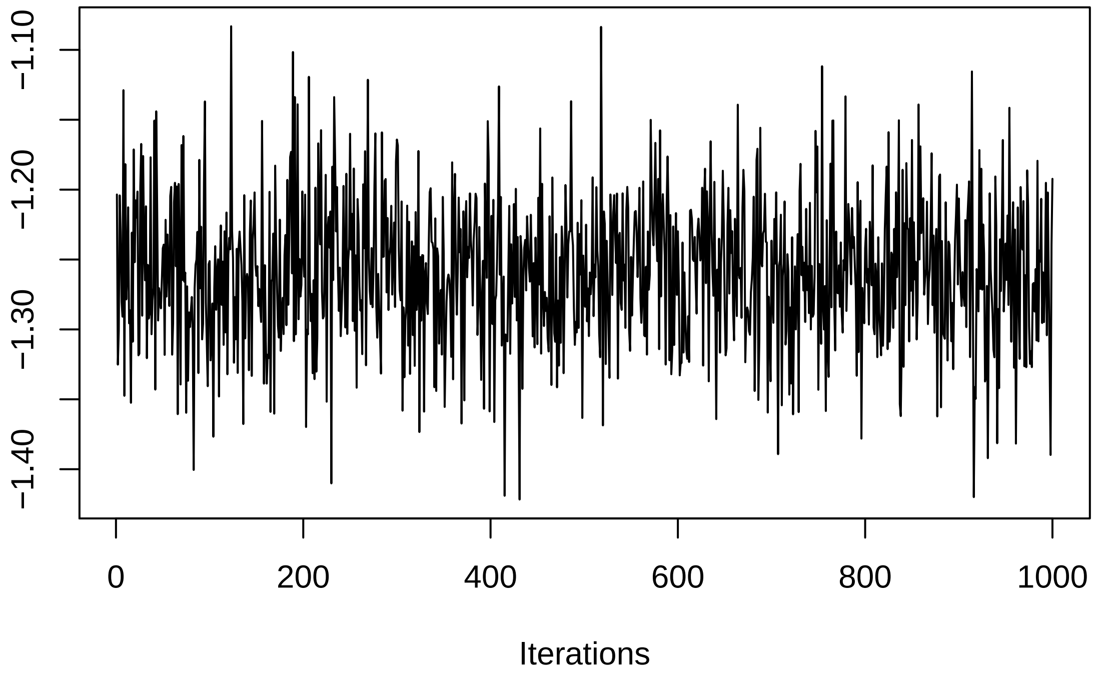
Extract the parameters and format them into an `mcmc` object.

```r
mcmcParamX <- as.mcmc(spClimate, parameters = "paramX")
mcmcMeansParamX <- as.mcmc(spClimate, parameters = "meansParamX")
mcmcVarX <- as.mcmc(spClimate, parameters = "varX")
```

Visually

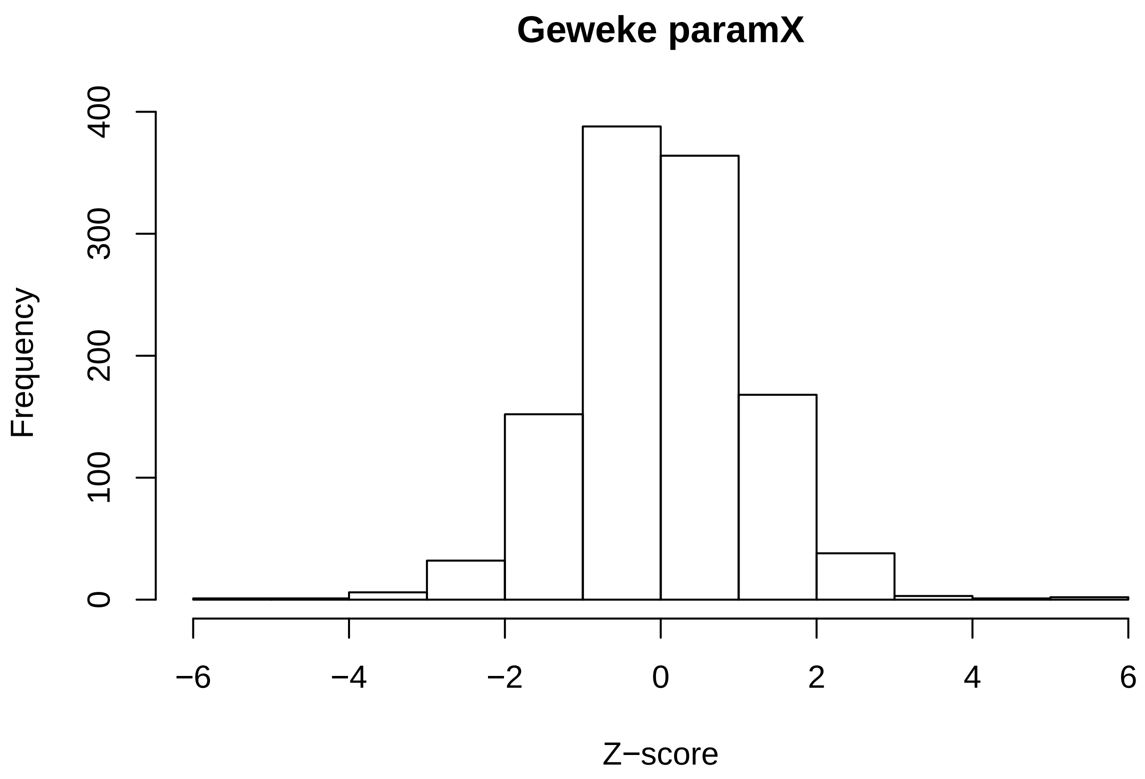
With the number of parameters, this can be long and tedious

```
traceplot(mcmcMeansParamX[,1])
```

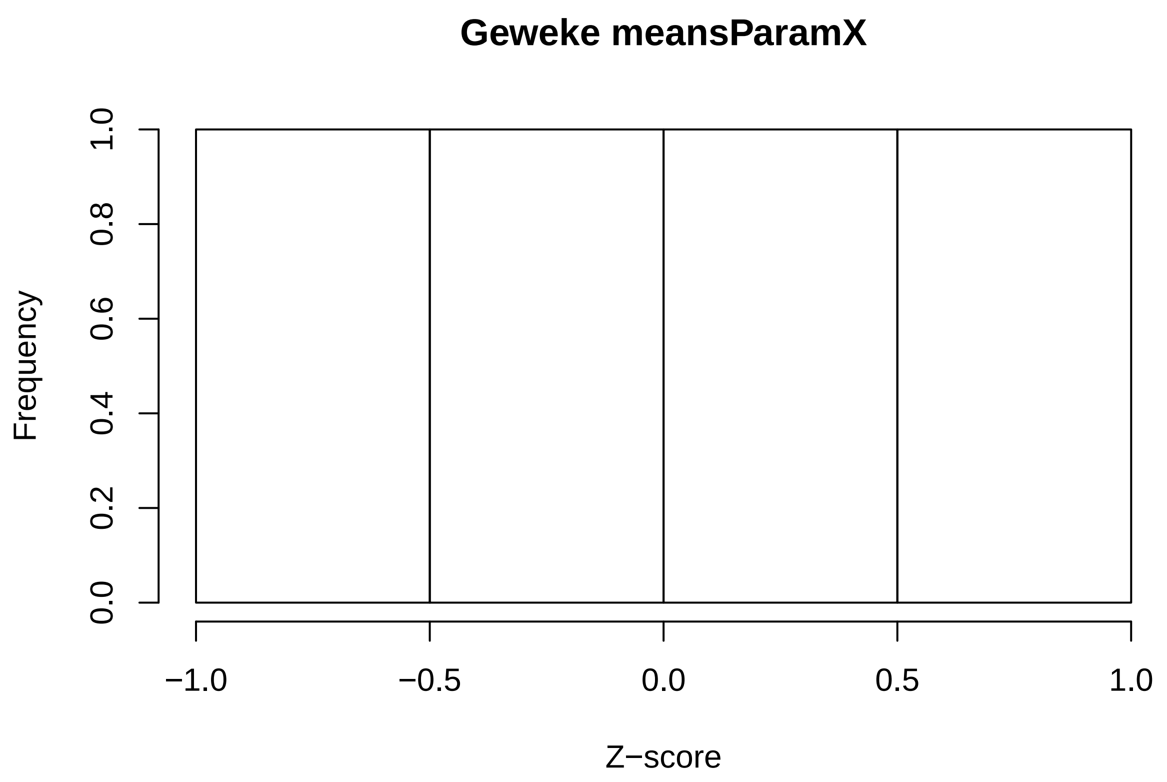


Geweke diagnostic

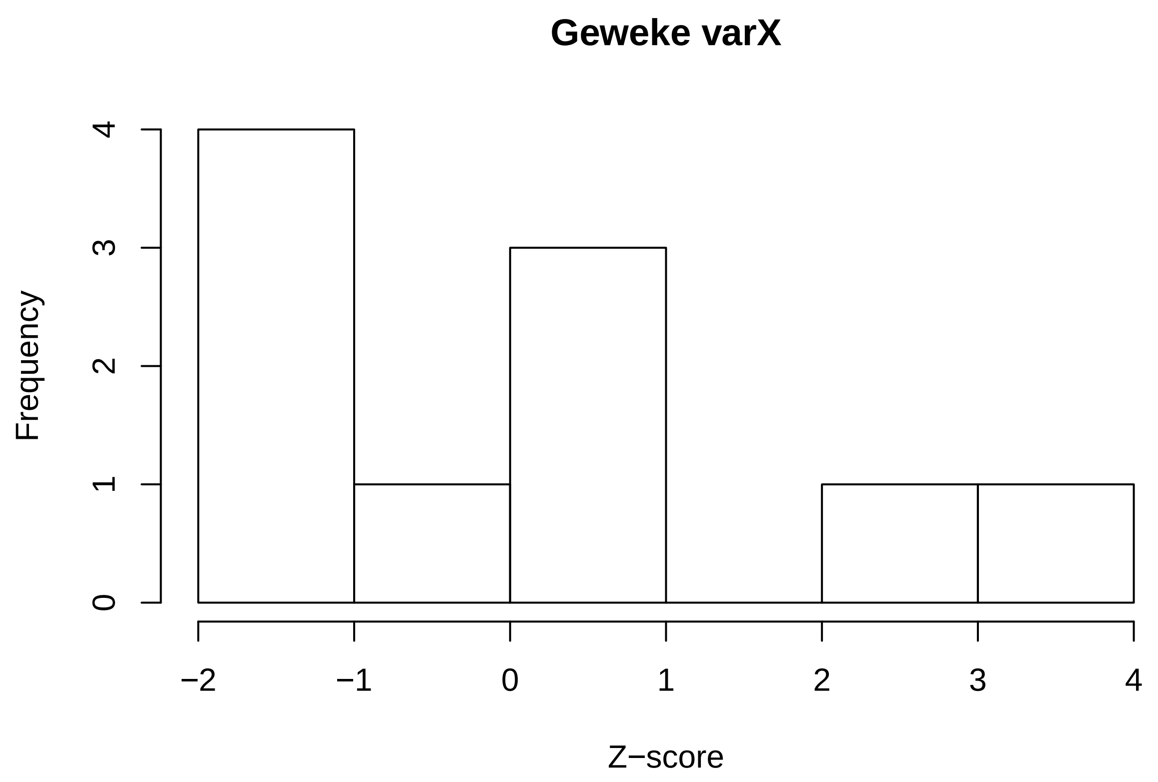
```
gewekeParamX <- geweke.diag(mcmcParamX,
                            frac1 = 0.5,
                            frac2 = 0.5)
hist(gewekeParamX$z,
     xlab = "Z-score",
     main = "Geweke paramX")
```



```
gewekeMeansParamX <- geweke.diag(mcmcMeansParamX,
                                 frac1 = 0.5,
                                 frac2 = 0.5)
hist(gewekeMeansParamX$z,
     xlab = "Z-score",
     main = "Geweke meansParamX")
```



```r
gewekeVarX <- geweke.diag(mcmcVarX,
                          frac1 = 0.5,
                          frac2 = 0.5)
hist(gewekeVarX$z,
     xlab = "Z-score",
     main = "Geweke varX")
```



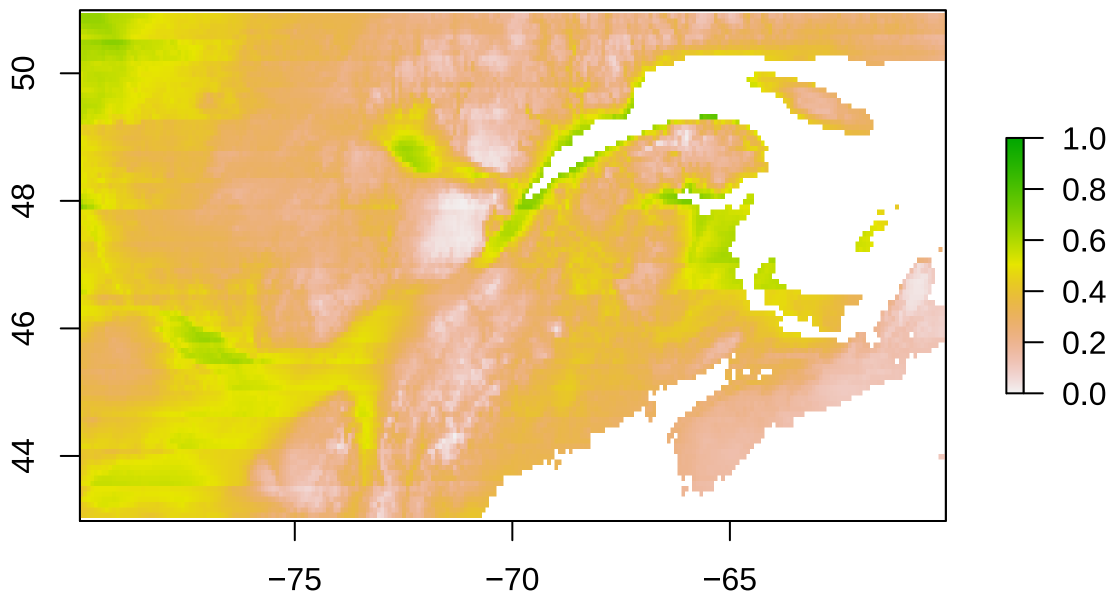
The MCMC seemed to have converged properly.

Prediction map

```r
formPredData <- as.HMSCdata(X = climateAxes)
pred <- predict(spClimate,
                newdata = formPredData,
                type = "response")
```

```r
spPredRaster <- brick(climatePresent, nl = ncol(spDat))
values(spPredRaster)[-NALoc,] <- pred

plot(spPredRaster[[1]], zlim = c(0,1))
```



50
48
46
44
−75
−70
−65
1.0
0.8
0.6
0.4
0.2
0.0