Supplement of Web Ecol., 24, 47–57, 2024
https://doi.org/10.5194/we-24-47-2024-supplement
© Author(s) 2024. CC BY 4.0 License.

Web Ecology Open Access

Supplement of

Towards spatial predictions of disease transmission risk: classical scrapie spill-over from domestic small ruminants to wild cervids

Nuno Mouta et al.

Correspondence to: Nuno Mouta (nuno.mouta@cibio.up.pt)

The copyright of individual parts of the supplement might differ from the article licence.

```
## ------biomod2-Based-on-Spectral-information-data------------ ##
## NOTES:
## (i) This version allows to use either true train absences or
## pseudo-absences by modifying the parameter useWhat
##
## (ii) If using pseudo-absences, the number of pseudo-absences
## can be modified using the parameter numberOfPseudoAbs (default:
## is equal to the number of presences)
##
## ------Please-have-the-fallowing-packages-installed--------##

library(biomod2)
library(raster)
library(sf)
library(fasterize)
library(dplyr)
library(sp)

convertToGeoTIFF <- function(inputFolder,outputFolder){

 fl <- list.files(inputFolder, pattern= ".grd$", full.names = TRUE)

 if(!dir.exists(outputFolder)){
  dir.create(outputFolder)
 }

 for(f in fl){

  cat("Converting file:/n",f," .../n",sep="")
  fn <- tools::file_path_sans_ext(basename(f))
  raster::stack(f) %>%
   raster::writeRaster(filename = paste(outputFolder,"/",fn,".tif",sep=""))
  cat("done!/n/n")
 }
}
# Set working directory
setwd("C:/.../...")

# Train polygons from shapefile
shapeTrain <- "C:/.../.shp"

trainPolygons <- read_sf(shapeTrain)

# Use train absences OR pseudo-absences
useWhat <- " pseudo-absences " # chose between "pseudo-absences  ,  train absences  "

# Input satellite images from folder
inputDirSatImages <- "C:/.../..."

imgList <- list.files(inputDirSatImages, pattern=".tif$", full.names = TRUE)

# Make a raster stack of satellite images
```

```
satImages <- stack(imgList)

names(satImages) <- substr(names(satImages),1,12)

# Convert train polygons to raster, where polygons with 0 represents absence and with 1 presence, please define the field containing this information as Id
rst <- fasterize(trainPolygons, satImages[[1]], field = "Id")

# Create xy data from raster
xyCoords <- xyFromCell(rst, 1:ncell(rst))
rstValues <- values(rst)

trainDF <- as.data.frame(cbind(xyCoords, rstValues))
colnames(trainDF) <- c("x","y","Id")

# Set the number of pseudo-absences equal to presence pixels
numberOfPseudoAbs <- nrow(trainDF %>% dplyr::filter(Id == 1))

# Specific values to test can be used like:
# numberOfPseudoAbs <- 500

#Identification of the work
spName <- "study_name"

## ------------------------------------------------------------------------------------ ##
## Data formatting and preparation ----
## ------------------------------------------------------------------------------------ ##

if(useWhat == "pseudo-absences"){

  # Extract only positive train points and select only the XY cols
  trainData <- trainDF %>%
    dplyr::filter(Id == 1) %>%
    dplyr::select(x, y)

  # Convert train data to spatial points object
  spPointsTrain <- SpatialPoints(trainData)

  myBiomodData <- BIOMOD_FormatingData(resp.var = spPointsTrain,
                                       expl.var = satImages,
                                       resp.name = spName,
                                       PA.nb.rep = 3, # Number of Pseudo-Absences sets
                                       PA.nb.absences = numberOfPseudoAbs,
                                      PA.strategy = 'random') # PA generation method

}else if(useWhat == "train absences"){

  # Select train data by Id (either 0's OR 1's i.e. true absences and true presences)
  trainData <- trainDF %>% dplyr::filter(Id == 0 | Id == 1)
  # Get XY coordinates
  spPointsTrain <- as.matrix(trainData[,c("x","y")])
```

```
  myBiomodData <- BIOMOD_FormatingData(resp.var = trainData$Id, # 0, 1 input vector
                                       resp.xy = spPointsTrain, # XY coords for 0's and 1's
                                       expl.var = satImages,
                                       resp.name = spName) # PA generation method
}else{
  stop("Wrong option in useWhat parameter! Must be pseudo-absences OR train absences.")
}

## ------------------------------------------------------------------------------------- ##
## Calibrate models ----
## ------------------------------------------------------------------------------------- ##

myBiomodModelOut <- BIOMOD_Modeling(
  data = myBiomodData, # Input data
  models = c('RF', 'CTA', 'ANN', 'GLM', 'GAM'), # Models to run
  #models.options =
  NbRunEval = 10, # Number of Evaluation runs - default 10
  DataSplit = 80,
  Prevalence = 0.5, # Prevalence between 0 and 1
  VarImport = 5, # Nr of rounds to evaluate variables - default 5
  models.eval.meth = c('TSS','ROC','KAPPA'), # Evaluation metrics
  SaveObj = TRUE,
  rescal.all.models = FALSE,
  do.full.models = TRUE)

# Save calibration object
saveRDS(myBiomodModelOut, paste(spName,"_myBiomodModelOut.RDS",sep=""))

# Get model evaluation values
myBiomodModelEval <- get_evaluations(myBiomodModelOut)

# Print ROC scores
print(myBiomodModelEval["ROC","Testing.data",,,])
print(myBiomodModelEval["TSS","Testing.data",,,])

# Get boxplot stats
print(fivenum(as.numeric(myBiomodModelEval["ROC","Testing.data",,,])))
print(fivenum(as.numeric(myBiomodModelEval["TSS","Testing.data",,,])))

# Save evaluation metrics from the arrays
evalDF.ROC <- as.data.frame(myBiomodModelEval["ROC","Testing.data",,,])
evalDF.TSS <- as.data.frame(myBiomodModelEval["TSS","Testing.data",,,])
evalDF.KAPPA <- as.data.frame(myBiomodModelEval["KAPPA","Testing.data",,,])

write.csv(evalDF.ROC, file =
paste(getwd(),"/",spName,"/",spName,"_evalDF_ROC.csv",sep=""))
write.csv(evalDF.TSS, file = paste(getwd(),"/",spName,"/",spName,"_evalDF_TSS.csv",sep=""))
write.csv(evalDF.KAPPA, file =
paste(getwd(),"/",spName,"/",spName,"_evalDF_KAPPA.csv",sep=""))

# Calculate variable importance across all PA sets, eval rouns and algorithms
varImportance <- get_variables_importance(myBiomodModelOut)
```

```
varImportanceByVariableAVG <- apply(varImportance,1,mean, na.rm=TRUE)
varImportanceByVariableSTD <- apply(varImportance,1,sd, na.rm=TRUE)
vimpDF <- data.frame(cnames    = names(varImportanceByVariableAVG),
                vimpAVG   = varImportanceByVariableAVG,
                varImpSTD = varImportanceByVariableSTD) %>%
 arrange(desc(vimpAVG))

print(vimpDF)
write.csv(vimpDF, file = paste(getwd(),"/",spName,"/",spName,"_varImportance.csv",sep=""))

## ------------------------------------------------------------------------------------ ##
## Perform ensemble modelling ----
## ------------------------------------------------------------------------------------ ##

myBiomodEM <- BIOMOD_EnsembleModeling(modeling.output = myBiomodModelOut,
                          #chosen.models = selMods,
                          em.by = 'all',
                          eval.metric = c('TSS'),
                          eval.metric.quality.threshold = 0.8,
                          prob.mean = TRUE,
                          prob.mean.weight.decay = 'proportional')

saveRDS(myBiomodEM, paste(spName,"_myBiomodEM.RDS",sep=""))

# Models to consider in the ensemble and projection
modelsToUse <- get_kept_models(myBiomodEM, 1)

saveRDS(modelsToUse, paste(spName,"_modelsToUse.RDS",sep=""))

# Get evaluation scores for the Ensemble Modelling stage
emEvalDF <- as.data.frame(get_evaluations(myBiomodEM))
write.csv(emEvalDF, file =
paste(getwd(),"/",spName,"/",spName,"_EnsMod_evalDF_AllMetrics.csv",sep=""))

# Obtain projections
myBiomodProj <- BIOMOD_Projection(modeling.output = myBiomodModelOut,
                        new.env = satImages,
                        proj.name = "Current", ## Name of the projection from above variable
proj.name
                        selected.models = modelsToUse,
                        filtered.meth = NULL,
                        binary.meth = NULL,
                        compress = 'gzip',
                        clamping.mask = TRUE,
                        output.format = '.grd',
                        do.stack = TRUE)

# Perform the ensembling of projections
myBiomodEF <- BIOMOD_EnsembleForecasting(projection.output = myBiomodProj,
                             binary.meth = c('TSS','ROC','KAPPA'),
                             EM.output = myBiomodEM,
                             output.format = '.grd')
```

```
# Perform the ensembling of projections
myBiomodEF <- BIOMOD_EnsembleForecasting(projection.output = myBiomodProj,
                                         binary.meth = c('TSS','ROC','KAPPA'),
                                         EM.output = myBiomodEM,
                                         output.format = '.tiff')

# Convert all output raster files to GeoTIFF
inFolder <- paste(getwd(),"/",spName,"/proj_",projName,sep="")
outFolder <- paste(inFolder,"/","GeoTIFF", sep="")
dir.create(outFolder)

#Save GeoTIFF(inFolder, outFolder)

save.image(file=paste(spName,"ModObjects-v1.RData",sep="_"))
```