Supplementary Data

Mediterranean Marine Science
Indexed in WoS (Web of Science, ISI Thomson) and SCOPUS
The journal is available on line at http://www.medit-mar-sc.net
www.hcmr.gr
DOI: http://doi.org/10.12681/mms.39523

Contribution to the Special Issue: Marine Animal Forest of the World (MAF WORLD)

Licence to predict – Investigating approaches to modelling low-occurrence deep-sea Irish Antipatharia with a new evaluation metric

Alexa PARIMBELLI, Mark P. JOHNSON, Kerry HOWELL, Claire LAGUIONIE MARCHAIS, and A. Louise ALLCOCK

Mediterranean Marine Science, 26, 2 (special issue) 2025

Section B

Modelling using 'gbm' package

```
# The code is a modification of the tutorial found in Elith et al. 2008, Appendix S3.

#### --- STEP 1 --- ####
library("gbm")
library("Metrics")
library("dplyr")
library("verification")

# Load your data
setwd("path/to/your/directory")
eval.data <- read.csv("evaluation_data.csv", as.is=T)

setwd("path/to/your/directory")
model.data <- read.csv("gridded_occurrence_data.csv", as.is=T)

# Ensure that predictors have the same name in model.data and eval.data # Ensure that brt.functions.R (available from Elith et al. 2008, Appendix S3) is located in the current working directory.
source("brt.functions.R")

#### --- STEP 2 --- ####
#Create a directory for variable selection process (only for approach using parsimonious variable selection)

dir.create("variable_selection")
setwd("variable_selection")

library("gbm.auto")
library("devtools")

#Find optimal bag fraction
#samples = dataset, resvar = column with response variable
gbm.bfcheck(samples = model.data, resvar = 1)

#Set a list of variables for each starting variable combination
variable_combinations <- list(
  TC = c(5, 6, 10, 12, 13, 14, 15),
  TFP = c(5, 7, 8, 10, 12, 13, 14, 15),
  TPP = c(5, 8, 9, 10, 12, 13, 14, 15))
```

```
# List to store the trained models
trained_models <- list()

# Loop through variable combinations
for (var_name in names(variable_combinations)) {
  model_name <- paste(var_name, "your_species_selvar", sep = "_")
    gbm_model <- gbm.step(data = model.data,
                          gbm.x = variable_combinations[[var_name]],
                          gbm.y = 1,
                          family = "bernoulli",
                          tree.complexity = 2,
                          learning.rate = 0.01,
                          bag.fraction = 0.75)

  # Store the trained model in the list with the desired model name
  trained_models[[model_name]] <- gbm_model
}

#### --- STEP 3 --- ####
## Calculate AIC and AICc for each variable combination

# Calculate the AICc for the 'TC' model
# Access the 'TC' model from the list of trained models
TC_model <- trained_models[["TC_your_species_selvar"]]
TC_log_likelihood <- sum(log(predict(TC_model, newdata = model.data, type = "response")))
print(TC_log_likelihood)
TC_num_trees <- TC_model$n.trees
TC_num_parameters <- TC_num_trees * (interaction_depth + 1)
TC_AIC <- -2 * TC_log_likelihood + 2 * TC_num_parameters
TC_AICc <- TC_AIC + (2 * TC_num_parameters * (TC_num_parameters + 1)) / (num_observations - TC_num_parameters - 1)

# Repeat for remaining combinations

#Create .csv to store these results into
# Assuming 'trained_models' is your list of trained GBM models
# Assuming 'model.data' is your training data
# Assuming 'interaction_depth' is a constant value
interaction_depth <- 2
num_observations <- nrow(model.data)

# Initialize an empty dataframe to store the results
result_df <- data.frame(
  Variable_Combination = character(length(trained_models)),
  n_observations = rep(num_observations, length(trained_models)),
  n_trees = numeric(length(trained_models)),
  tree_complexity = rep(interaction_depth, length(trained_models)),
  n_parameters = numeric(length(trained_models)),
  log_likelihood = numeric(length(trained_models)),
  AIC = numeric(length(trained_models)),
  AICc = numeric(length(trained_models))
)

# Loop through each model to calculate and store the required values
for (i in seq_along(trained_models)) {
  model_name <- names(trained_models)[i]
  current_model <- trained_models[[i]]

  # Calculate log-likelihood
  log_likelihood <- sum(log(predict(current_model, newdata = model.data, type = "response")))
```

```
  # Calculate number of trees, number of parameters, AIC, and AICc
  n_trees <- current_model$n.trees
  n_parameters <- n_trees * (interaction_depth + 1)
  AIC <- -2 * log_likelihood + 2 * n_parameters
  AICc <- AIC + (2 * n_parameters * (n_parameters + 1)) / (num_observations - n_parameters - 1)

  # Store the calculated values in the dataframe
   result_df[i, ] <- c(model_name, num_observations, n_trees, interaction_depth, n_parameters, log_likelihood, AIC,
AICc)
}

# Set column names
colnames(result_df) <- c("Variable_Combination", "n_observations", "n_trees", "tree_complexity", "n_parameters",
"log_likelihood", "AIC", "AICc")

# Display the resulting dataframe
print(result_df)

#### --- STEP 4 --- ####
## Calculate AUC for each variable combination

## Write TC final model
setwd("path/to/folder/variable_selection")
dir.create("TC")
setwd("TC")

library("gbm")
TC_full_preds <- predict.gbm(TC_model,
n.trees=TC_model$gbm.call$best.trees,
type="response")
library("dplyr")
TC_full_mod_csv <- bind_cols(model.data, TC_full_preds)
head(TC_full_mod_csv)
colnames(TC_full_mod_csv)[16] <- "preds"
head(TC_full_mod_csv)
#Add species column
TC_full_mod_csv$species <- "your_species"
head(TC_full_mod_csv)
#Writing.csv
TC_full_mod_csv <- TC_full_mod_csv[c('species', 'PA', 'preds', 'Xcoord', 'Ycoord')]
head(TC_full_mod_csv)
write.csv(TC_full_mod_csv, "TC_Full_mod_preds.csv", sep=",")

# Calculate TC Deviance, ROC, AUC
TC_deviance <- calc.deviance(TC_full_mod_csv$PA, TC_full_mod_csv$preds,calc.mean=T) #Calculates deviance,
NB default is "binomial"
verification::roc.plot(TC_full_mod_csv$PA,
                       TC_full_mod_csv$preds,
                       xlab = "True positive rate (TPR)",
                       ylab = "False positive rate (FPR)")

TC_AUC <- roc(TC_full_mod_csv$PA, TC_full_mod_csv$preds) #Calculates AUC

TC_RMSE <- rmse(TC_full_mod_csv$PA, TC_full_mod_csv$preds)

# Repeat for the remaining combinations

# Adding a new column using square bracket notation
result_df[["deviance"]] <- c(TC_deviance, TFP_deviance, TPP_deviance)
```

```
result_df[["AUC"]] <- c(TC_AUC, TFP_AUC, TPP_AUC)
result_df[["RMSE"]] <- c(TC_RMSE, TFP_RMSE, TPP_RMSE)

# Display the updated dataframe
print(result_df)


#### --- STEP 5 --- #### -
# Find and remove the least important variable

# Access summary of the model with the variable combination that obtained the lowest absolute AICc.
model_summary <- summary(lowest_AICc_model)
# Extract relative influence of variables
var_importance <- model_summary$rel.inf
print(var_importance)

# Create a data frame
var_importance_df <- data.frame(
                Var1 = var_importance[1],
                Var2 = var_importance[2],
                Var3 = var_importance[3],
                #etc..)

# Print the data frame
print(var_importance_df)
write.csv(var_importance_df, "var_importance_lowest_AICc_model.csv")


#Run the model without least important variable
New_model <- gbm.step(data=model.data,
                 gbm.x = c(5, 7, 8, 10, 13, 14, 15),
                 gbm.y = 1,
                 family = "bernoulli",
                 tree.complexity = 2,
                 bag.fraction = 0.75)

setwd("path/to/folder/variable_selection")
dir.create("New_model")
setwd("New_model")



#Calculate AIC and AICc
New_model_log_likelihood <- sum(log(predict(New_model_model, newdata = model.data, type = "response")))
print(New_model_log_likelihood)
New_model_num_trees <- New_model_model$n.trees
New_model_num_parameters <- New_model_num_trees * (interaction_depth + 1)
New_model_AIC <- -2 * New_model_log_likelihood + 2 * New_model_num_parameters
New_model_AICc <- New_model_AIC + (2 * New_model_num_parameters * (New_model_num_parameters + 1))
/ (num_observations - New_model_num_parameters - 1)


#Write .csv with predictions
New_model_full_preds <- predict.gbm(New_model_model, n.trees=New_model_model$gbm.call$best.trees,
type="response")
New_model_full_mod_csv <- bind_cols(model.data, New_model_full_preds)
head(New_model_full_mod_csv)
colnames(New_model_full_mod_csv)[16] <- "preds"
head(New_model_full_mod_csv)
#Add species column
```

```
New_model_full_mod_csv$species <- "your_species"
head(New_model_full_mod_csv)
#Writing.csv
New_model_full_mod_csv <- New_model_full_mod_csv[c('species', 'PA', 'preds', 'Xcoord', 'Ycoord')]
write.csv(New_model_full_mod_csv, "New_model_Full_mod_preds.csv", row.names = FALSE)

# Calculate TC Deviance, ROC, AUC
New_model_deviance <- calc.deviance(New_model_full_mod_csv$PA, New_model_full_mod_csv$preds,calc.
mean=T) verification::roc.plot(New_model_full_mod_csv$PA,
                    New_model_full_mod_csv$preds,
                    xlab = "True positive rate (TPR)",
                    ylab = "False positive rate (FPR)")
New_model_AUC <- roc(New_model_full_mod_csv$PA, New_model_full_mod_csv$preds)
New_model_RMSE <- rmse(New_model_full_mod_csv$PA, New_model_full_mod_csv$preds)

# Add new row with all the values
new_row <- data.frame(
  Variable_Combination = "New_model_your_species_selvar",
  n_observations = num_observations,
  n_trees = New_model_num_trees,
  tree_complexity = interaction_depth,
  n_parameters = New_model_num_parameters,
  log_likelihood = -New_model_log_likelihood,
  AIC = New_model_AIC,
  AICc = New_model_AICc,
  deviance = New_model_deviance,
  AUC = New_model_AUC,
  RMSE = New_model_RMSE
)
# Add the new row to the existing dataframe
result_df <- rbind(result_df, new_row)

# Display the updated dataframe
print(result_df)

# Access summary of the model
model_summary <- summary(New_model_model)
# Extract relative influence of variables
var_importance <- model_summary$rel.inf
print(var_importance)

# Create a data frame
var_importance_df <- data.frame(
  VelX = var_importance[1],
  VelY = var_importance[2],
  Temp = var_importance[3],
  FBPI = var_importance[4],
  PlanCurv = var_importance[5],
  Rug = var_importance[6],
  BBPI = var_importance[7])
# Print the data frame
print(var_importance_df)
write.csv(var_importance_df, "var_importance_New_model.csv")

# Iterate the process until the variable combination producing the lowest AICc is found.

#### --- STEP 6 --- ####
#TUNE HYPERPARAMETERS
#Try with different learning rates - gbm.step finds the optimal number of trees for the given bag fraction and learning
rate
```

```
#Optimal bag fraction is calculated by bf.check (Dedman et al. 2017)
#We aim for a number of trees higher than 1000 (Elith et al. 2008)

#Calculate the AICc and AUC for each model and use AICc as stopping rule
setwd("/path/to/directory")
dir.create("lr_selection")

# Repeat for LR = 0.1, 0.05, 0.01, 0.005, 0.001

# Example for LR = 0.1
setwd("path/to/folder/lr_selection")
dir.create("LR_0.1")
setwd("LR_0.1")

LR_0.1_model <- gbm.step(data = model.data,
                         gbm.x = c(5, 7, 8, 10, 13, 14, 15),
                         gbm.y = 1,
                         family = "bernoulli",
                         tree.complexity = 2,
                         learning.rate = 0.1,
                         bag.fraction = 0.75)

# Calculate AIC
LR_0.1_log_likelihood <- sum(log(predict(LR_0.1_model, newdata = model.data, type = "response")))
print(LR_0.1_log_likelihood)
LR_0.1_num_trees <- LR_0.1_model$n.trees
LR_0.1_num_parameters <- LR_0.1_num_trees * (interaction_depth + 1)
LR_0.1_AIC <- -2 * LR_0.1_log_likelihood + 2 * LR_0.1_num_parameters
LR_0.1_AICc <- LR_0.1_AIC + (2 * LR_0.1_num_parameters * (LR_0.1_num_parameters + 1)) / (num_observa-
tions - LR_0.1_num_parameters - 1)

# Write CSV file with model predictions
LR_0.1_full_preds <- predict.gbm(LR_0.1_model, n.trees=LR_0.1_model$gbm.call$best.trees, type="response")
LR_0.1_full_mod_csv <- bind_cols(model.data, LR_0.1_full_preds)
head(LR_0.1_full_mod_csv)
colnames(LR_0.1_full_mod_csv)[17] <- "preds"
head(LR_0.1_full_mod_csv)
LR_0.1_full_mod_csv$species <- "your_species"
head(LR_0.1_full_mod_csv)
LR_0.1_full_mod_csv <- LR_0.1_full_mod_csv[c('species', 'PA', 'preds', 'Xcoord', 'Ycoord')]
head(LR_0.1_full_mod_csv)
write.csv(LR_0.1_full_mod_csv, "LR_0.1_Full_mod_preds.csv", row.names = FALSE)

# Calculate deviance, AUC, RMSE
LR_0.1_deviance <- calc.deviance(LR_0.1_full_mod_csv$PA, LR_0.1_full_mod_csv$preds,calc.mean=T)

verification::roc.plot(LR_0.1_full_mod_csv$PA,
                       LR_0.1_full_mod_csv$preds,
                       xlab = "True positive rate (TPR)",
                       ylab = "False positive rate (FPR)")
LR_0.1_AUC <- roc(LR_0.1_full_mod_csv$PA, LR_0.1_full_mod_csv$preds)

LR_0.1_RMSE <- rmse(LR_0.1_full_mod_csv$PA, LR_0.1_full_mod_csv$preds)

# Create dataframe for all parameters
new_row <- data.frame(
  Variable_Combination = "LR_0.1_your_species_selvar",
  n_observations = num_observations,
  n_trees = LR_0.1_num_trees,
  tree_complexity = interaction_depth,
```

```
  n_parameters = LR_0.1_num_parameters,
  log_likelihood = -LR_0.1_log_likelihood,
  AIC = LR_0.1_AIC,
  AICc = LR_0.1_AICc,
  deviance = LR_0.1_deviance,
  AUC = LR_0.1_AUC,
  RMSE = LR_0.1_RMSE)
result_df <- rbind(result_df, new_row)
print(result_df)

#### --- STEP 7 --- ####
#Iterate the process until the model with the lowest AICc is found.
# E.g. if the models achieving the lowest AICc are the ones for LR=0.01 and LR = 0.005, calculate the midpoint between 0.1 and 0.005.
midpoint <- (0.025 + 0.02) / 2
print(midpoint)

# Calculate the model for the midpoint, e.g. LR = 0.013
LR_0.013_model <- gbm.step(data = model.data,
                           gbm.x = c(5, 7, 8, 10, 13, 14, 15),
                           gbm.y = 1,
                           family = "bernoulli",
                           tree.complexity = 2,
                           learning.rate = 0.013,
                           bag.fraction = 0.75)

setwd("path/to/folder/lr_selection")
dir.create("LR_0.013")
setwd("LR_0.013")

#Calculate AIC and AICc
LR_0.013_log_likelihood <- sum(log(predict(LR_0.013_model, newdata = model.data, type = "response")))
print(LR_0.013_log_likelihood)
LR_0.013_num_trees <- LR_0.013_model$n.trees
LR_0.013_num_parameters <- LR_0.013_num_trees * (interaction_depth + 1)
LR_0.013_AIC <- -2 * LR_0.013_log_likelihood + 2 * LR_0.013_num_parameters
LR_0.013_AICc <- LR_0.013_AIC + (2 * LR_0.013_num_parameters * (LR_0.013_num_parameters + 1)) / (num_observations - LR_0.013_num_parameters - 1)

# Write CSV with model predictions
LR_0.013_full_preds <- predict.gbm(LR_0.013_model, n.trees=LR_0.013_model$gbm.call$best.trees, type="response")
LR_0.013_full_mod_csv <- bind_cols(model.data, LR_0.013_full_preds)
head(LR_0.013_full_mod_csv)
colnames(LR_0.013_full_mod_csv)[16] <- "preds"
head(LR_0.013_full_mod_csv)
LR_0.013_full_mod_csv$species <- "your_species"
head(LR_0.013_full_mod_csv)
LR_0.013_full_mod_csv <- LR_0.013_full_mod_csv[c('species', 'PA', 'preds', 'Xcoord', 'Ycoord')]
head(LR_0.013_full_mod_csv)
write.csv(LR_0.013_full_mod_csv, "LR_0.013_Full_mod_preds.csv", row.names = FALSE)

# Calculate deviance, AUC and RMSE
LR_0.013_deviance <- calc.deviance(LR_0.013_full_mod_csv$PA, LR_0.013_full_mod_csv$preds,calc.mean=T)
verification::roc.plot(LR_0.013_full_mod_csv$PA,
                       LR_0.013_full_mod_csv$preds,
                       xlab = "True positive rate (TPR)",
                       ylab = "False positive rate (FPR)")
LR_0.013_AUC <- roc(LR_0.013_full_mod_csv$PA, LR_0.013_full_mod_csv$preds)
LR_0.013_RMSE <- rmse(LR_0.013_full_mod_csv$PA, LR_0.013_full_mod_csv$preds)
```

```
# Add new row to the dataframe with parameters
new_row <- data.frame(
  Variable_Combination = "LR_0.013_your_species_selvar",
  n_observations = num_observations,
  n_trees = LR_0.013_num_trees,
  tree_complexity = interaction_depth,
  n_parameters = LR_0.013_num_parameters,
  log_likelihood = -LR_0.013_log_likelihood,
  AIC = LR_0.013_AIC,
  AICc = LR_0.013_AICc,
  deviance = LR_0.013_deviance,
  AUC = LR_0.013_AUC,
  RMSE = LR_0.013_RMSE)
result_df <- rbind(result_df, new_row)
print(result_df)

#### --- STEP 8 --- ####
# Write CSV with model results
setwd("path/to/directory")
write.csv(result_df, "AICc_Results_your_species.csv")

#### --- STEP 9 --- ####
#Print the best model
#your_best_model is the model that has the lowest AICc.
setwd("path/to/directory")
dir.create("bestmodel")
setwd("bestmodel")

# Use predict.gbm() to make predictions
library(gbm)
preds <- predict.gbm(Your_best_model,
eval.data[c(4:15)], # only select relevant columns
n.trees=Your_best_model$gbm.call$best.trees, type="response"
)

#Prepare data frame
head(preds)
library("dplyr")
library("raster")
df_addpred <- bind_cols(eval.data, preds)
colnames(df_addpred)[16] <- "preds"
head(df_addpred)
df <- df_addpred[c('Xcoord', 'Ycoord', 'preds')]
head(df)
r <- rasterFromXYZ(df)
writeRaster(r,'your_species.predictions_gbm.asc', format='ascii')
```

Manual assemblage of train and test data

```
# To use for MaxEnt and GBM predictions
# load presence only transect crib sheet (list of transect codes).
Transcrib <- read.csv("TransCrib.csv",head=TRUE,sep=",")

# Load data sheet for the species (For each cell, it contains presence/absence column, coordinates, transect code for each cell, and environmental variables)
Allobs <- read.csv("Allobs.csv", head=TRUE,sep=",")

###CAREFUL YOU NEED TO CHANGE a, x and y values below for each new species your are modelling
```

```
# Sample pres transects and neg transects (product is row numbers). This is an example where the modelled species was observed in 19 transects of the 53 explored.
a = 19 #number of presence transects
end = 53 #total number of transects
0.7*a
x = 13 #70% of a (should be rounded to the lowest entire number)
0.7*(end-a)
y = 23 #70% of absence transect = 0.7*(end-a)   also need to be rounded

#prevalence ratio to reach
presence = sum(allobs$PA != 0)    #number of PA = 1 in Allobs
absence = sum(allobs$PA == 0) #number of PA = 0 in Allobs
prev = 100*(presence/(presence +absence))
print("the prevalence ratio is")
prev

#here we sample the presence and absence transect (keepping row number of each file)
step1P<-sample(1:a,size=x,replace=FALSE)
step1A<-sample(((a+1):end),size=y,replace=FALSE)
# apply row numbers to transect crib sheet and row bind the P&A
step1Pss<-transcrib[step1P,]
step1Ass<-transcrib[step1A,]
step1ss<-rbind(step1Pss,step1Ass)

# merge prestrans with transect crib sheet retaining all the rows not selected in the ss
selecting<-merge(allobs,step1ss,by="Group_trans",all=TRUE)
# non-ss rows NA values (in the inclusion column) replaced with 0
selecting[is.na(selecting)]<-0

mat_calc = data.frame(selecting[,5],selecting[,18])
names(mat_calc)[names(mat_calc) == 'selecting...5.'] <- 'PA'
names(mat_calc)[names(mat_calc) == 'selecting...18.'] <- 'Train'

tot_test = sum(mat_calc[,2]== 0)
tot_train =  sum(mat_calc[,2]== 1)
pres_test = sum((mat_calc[,2]== 0)& (mat_calc[,1]== 1))
pres_train = sum((mat_calc[,2]== 1)& (mat_calc[,1]== 1))

#compute the prevalence ratio of train and test
Percent_test = pres_test*100/tot_test
print("The test ratio is")
Percent_test
Percent_train = pres_train*100/tot_train
print("The train ratio is")
Percent_train

#THIS PART BELOW CAN ONLY BE DONE IF THE PREVALENCE RATIO OF TRAIN AND TEST ARE -/+ 1% of the prevalence of Allobs
#if not redo until you find a correct model

#if ok, proceed

names(selecting)[names(selecting) == 'PA.x'] <- 'PA'
names(selecting)[18]<- 'Train'
names(selecting)[names(selecting) == 'PA.y'] <- 'Train'
selecting <- selecting[, c(2,3,4,5,1,6,7,8,9,10,11,12,13,14,15,16,17,18)]

W<-split(selecting, selecting$Train)
Testfile = W[[1]]
```

```
Trainfile = W[[2]]

U <-split(Trainfile, Trainfile$PA)

#### NEED TO CHECK THAT 1 is PA = 0 and 2 is PA = 1, otherwise reverse
Train_abs = U[[1]]
Train_pres = U[[2]]

#Here CHANGE THE file number (increase i by 1 everytime you have data matching the criterion)

# Write data to csv, make sure to change number (01 to 10) every time
write.table(selecting, file = "selection01.csv", sep = ",",qmethod = "double",row.names = FALSE)
write.table(Testfile, file = "Test01.csv", sep = ",",qmethod = "double",row.names = FALSE)
write.table(Train_abs, file = "Train_abs_01.csv", sep = ",",qmethod = "double",row.names = FALSE)
write.table(Train_pres, file = "Train_pres_01.csv", sep = ",",qmethod = "double",row.names = FALSE)

## BINDING CODE TO GET FULL TRAINING DATASET ##
library("tidymodels")
setwd("path/to/folder/traintestpart/TrainTest01")

absence <- read.csv("Train_abs_10.csv", header = TRUE, sep = ",")
presence <- read.csv("Train_pres_10.csv", header = TRUE, sep = ",")

training <- bind_rows(absence, presence)
head(training)
write.csv(training, "Training10.csv")
```

Train/test in 'gbm' package

```
#### --- STEP 1 --- ####
#Example for one dataset (TrainTest01)
#NB: be sure that brt.functions.R is located in your working directory
library("gbm")
source("brt.functions.R")

#Load data
setwd("path/to/folder/traintestpart/TrainTest01")

model.data <- read.csv("Training01.csv", as.is=T) #Loads samples data
head(model.data) #Prints first 6 rows to check the dataf
eval.data <- read.csv("Test01.csv", as.is=T) #Loads samples data
head(eval.data) #Prints first 6 rows to check the data

#### --- STEP 2 --- ####
## FITTING THE MODEL
tot.presence <- sum(model.data$PA) #Returns total number of presences
print(paste("The total number of presence is", tot.presence))

#Check the optimal bag fraction for the training dataset
gbm.bfcheck(samples = model.data, resvar = 5)

#Find optimal LR and test learning rates based on the lowest AICc
TrainTest01_model <- gbm.step(data=model.data,
gbm.x = c(7,9,13,17),
gbm.y = 5,
family = "bernoulli",
tree.complexity = 2,
learning.rate = 0.01,   #e.g. 0.01
bag.fraction = 0.84)
```

```
# Calculate AIC
TrainTest01_log_likelihood <- sum(log(predict(TrainTest01_model , newdata = model.data, type = “response”)))
print(TrainTest01_log_likelihood)
TrainTest01_num_trees <- TrainTest01_model$n.trees
TrainTest01_num_parameters <- TrainTest01_num_trees * (interaction_depth + 1)
TrainTest01_AIC <- -2 * TrainTest01_log_likelihood + 2 * TrainTest01_num_parameters
TrainTest01_AICc <- TrainTest01_AIC + (2 * TrainTest01_num_parameters * (TrainTest01_num_parameters + 1)) /
(num_observations - TrainTest01_num_parameters - 1)

#### --- STEP 3 --- ####
## PREDICTING TO TRAIN DATA
# Use predict.gbm() to make predictions
#Here we leave eval.data blank so that we can predict to training data
library(gbm)
train_preds <- predict.gbm(TrainTest01_model, n.trees = TrainTest01_model$gbm.call$best.trees, type=”response”)

#Preparing .csv file
library(“dplyr”)
train_mod <- bind_cols(model.data, train_preds)
colnames(train_mod)[20] <- “preds”
colnames(train_mod)[2] <- “species”
head(train_mod)

#Writing.csv
train_mod_csv <- train_mod[c(‘species’, ‘PA’, ‘preds’, ‘Xcoord’, ‘Ycoord’)]
head(train_mod_csv)
write.csv(train_mod_csv, “Training_mod_01_preds.csv”)
#### --- STEP 4 --- ####
## PREDICTING TO TEST DATA
# Use predict.gbm() to make predictions
library(“gbm”)
test_preds <- predict.gbm(TrainTest01_model, eval.data[c(6:17)], n.trees=TrainTest01_model$gbm.call$best.trees,
type=”response”)
head(eval.data)
#Preparing .csv file
library(“dplyr”)
test_mod <- bind_cols(eval.data, test_preds)
head(test_mod)
colnames(test_mod)[19] <- “preds”
colnames(test_mod)[1] <- “species”
head(test_mod)

#Writing.csv
test_mod_csv <- test_mod[c(‘species’, ‘PA’, ‘preds’, ‘Xcoord’, ‘Ycoord’)]
head(test_mod_csv)
write.csv(test_mod_csv, “Test_mod_01_preds.csv”)

#### --- STEP 5 --- ####
#Prepare raster
#Predict to environmental data
setwd(“path/to/folder”)

environment.data <- read.csv(“environmentdata.csv”, header = TRUE)
head(environment.data)

setwd(“path/to/folder/traintestpart/TrainTest01”)

env_preds <- predict.gbm(TrainTest01_model,
environment.data[c(4:15)],
```

```
n.trees=TrainTest01_model$gbm.call$best.trees, type="response")

env_mod <- bind_cols(environment.data, env_preds)
head(env_mod)
colnames(env_mod)[16 ] <- "preds"

#Write ASCII file
library("raster")
df <- env_mod[c('Xcoord', 'Ycoord', 'preds')]
head(df)
r <- rasterFromXYZ(df)
writeRaster(r,'TrainTest01.asc', format='ascii', overwrite = TRUE)
```

MESS analysis in R

```
library("modEvA")

setwd("path/to/folder")
model.data <- read.csv("occurrence_data.csv")
environment.data <- read.csv("environment_data.csv")

# Perform a MESS analysis
# We are extrapolating models from a reduced portion of the NE Atlantic (our sampling sites) to a bigger area in the NE Atlantic:

names(model.data)  # variables are in columns 4:15
names(environment.data)  # variables are in columns 4:15

sampling.site <- subset(model.data, select = 4:15)
sampling.site <- model.data[c(16,15,14,13,12,11,10,9,8,7,19,18)]
environment <- subset(environment.data, select = 4:15)
coordinates <- subset(environment.data, select=2:3)

#Use ONLY the variables that were selected for the analysis
#Double check which variables you want
names(sampling.site)
names(environment)
#Subset datasets to keep only the wanted variables
sampling.site <- sampling.site[c(2,3,7,9,10,11,12)]
environment <- environment[c(2,3,7,9,10,11,12)]

#Perform mess analysis
mess <- MESS(V = sampling.site, P = environment)

# The function returns a data frame with the same column names as P plus a column named TOTAL, quantifying the similarity between each point in the projection dataset and those in the reference dataset.
# Negative values indicate localities that are environmentally dissimilar from the reference region.
# The last column, MoD, indicates which of the column names of P corresponds to the most dissimilar variable, i.e., the limiting factor or the variable that drives the MESS in that locality (Elith et al. 2010).

head(mess)
library(dplyr)
mess.coord <- cbind(coordinates,mess)
names(mess.coord)
novel.ascii <- mess.coord[c(1,2,10)]
head(novel.ascii)

library(raster)
r <- rasterFromXYZ(novel.ascii)

setwd("path/to/folder")
writeRaster(r,'yourspecies_novel.asc', format='ascii', overwrite = TRUE)
```

Computing optimal thresholds

```
# The following code is adapted for MaxEnt models; for GBM models, change 'Cloglog' with preds, and modify the number of columns accordingly.
#### --- STEP 1 --- ####
## To compile the train/test datasets
library(dplyr)

setwd("path/to/folder/traintestpart/TrainTest01")
```

```
TF<-read.csv("Test01.csv",head=TRUE,sep=",")

#Creation of the training dataset
setwd("path/to/folder/traintestpart/TrainTest01")

BackPred<-read.csv("backgroundPredictions.csv",head=TRUE,sep=",")
new_df = BackPred
pa = integer(nrow(BackPred))
new_df$PA = pa
names(new_df)[5]<- 'Cloglog'

col_order <- c("x", "y", "PA", raw", "cumulative","Cloglog")
new_df2 <-new_df[, col_order]
SampPred<-read.csv("samplePredictions.csv",head=TRUE,sep=",")
names(SampPred)[1]<- 'x'
names(SampPred)[2]<- 'y'
names(SampPred)[3]<- 'PA'
names(SampPred)[4]<- 'raw'
names(SampPred)[5]<- 'cumulative'
names(SampPred)[6]<- 'Cloglog'

add_data = SampPred[SampPred$PA == "train", ]

new_df3 = rbind(new_df2,add_data)
new_df3[new_df3=="train"]<-1

new_df3$species=character(nrow(new_df3))
new_df3[new_df3==""]<-"yourspecies"

col_order2 <- c("species", "PA", "Cloglog", "x", "y", "raw", "cumulative")
new_df4 <-new_df3[, col_order2]

#Creation of the test dataset
SampPred2<-read.csv("samplePredictions.csv",head=TRUE,sep=",")
new_test_df =SampPred2
names(new_test_df)[1]<- 'x'
names(new_test_df)[2]<- 'y'
names(new_test_df)[3]<- 'PA'
names(new_test_df)[4]<- 'raw'
names(new_test_df)[5]<- 'cumulative'
names(new_test_df)[6]<- 'Cloglog'
new_test_df2 = new_test_df[new_test_df$PA == "test", ]
new_test_df3 = arrange(new_test_df2, x,y)
```

```
names(TF)[1]<- ‘species’
names(TF)[2]<- ‘x’
names(TF)[3]<- ‘y’
names(TF)[4]<- ‘PA’
TF2 = arrange(TF, x,y)

new_test_df3$PA = TF2$PA
new_test_df3$species=character(nrow(new_test_df3))
new_test_df3[new_test_df3==””]<-”yourspecies_m3”

col_order3 <- c(“species”, “PA”, “Cloglog”, x”, “y”, “raw”, “cumulative”)
new_test_df4 <-new_test_df3[, col_order3]

# Save datasets
setwd(“path/to/folder”)
dir.create(“presenceabsence”)
setwd(“presenceabsence”)

write.csv(new_df4, “Training_mod_01_preds.csv”)
write.csv(new_test_df4, “Test_mod_01_preds.csv”)

#### --- STEP 2 --- ####
## To compile the full model dataset

setwd(“path/to/optimised/model”)
BestPred < read.csv(“backgroundPredictions.csv”,head=TRUE,sep=”,”)
new_best_df = BestPred
pa_best = integer(nrow(BestPred))
new_best_df $PA = pa_best

col_order <- c(“x”, “y”, “PA”, “raw”, “cumulative”,”Cloglog”)
new_best_df2 <-new_best_df[, col_order]

SampBest<-read.csv(“samplePredictions.csv”,head=TRUE,sep=”,”)
names(SampBest)[1]<- ‘x’
names(SampBest)[2]<- ‘y’
names(SampBest)[3]<- ‘PA’
names(SampBest)[4]<- ‘raw’
names(SampBest)[5]<- ‘cumulative’
names(SampBest)[6]<- ‘Cloglog’

new_best_df3 = rbind(new_best_df2,SampBest)
new_best_df3[new_best_df3==”train”]<-1

new_best_df3$species=character(nrow(new_best_df3))
new_best_df3[new_best_df3==””]<-”yourspecies”

col_order4 <- c(“species”, “PA”, “Cloglog”, “x”,  “y”, “raw”, “cumulative”)
new_best_df4<-new_best_df3[, col_order4]

setwd(“path/to/folder/presenceabsence”)
write.csv(new_best_df4, “Full_mod_preds.csv”)

#### --- STEP 3 --- ####
#load library
library(PresenceAbsence)
library(matrixStats)
library(ggplot2)
```

```
setwd("path/to/folder/presenceabsence")

#load your data
SPDATA_test_01<-read.csv("Test_mod_01_preds.csv", head=T, sep=",")
SPDATA_test_01<-SPDATA_test_01[,2:8]
SPDATA_test_02<-read.csv("Test_mod_02_preds.csv", head=T, sep=",")
SPDATA_test_02<-SPDATA_test_02[,2:8]
SPDATA_test_03<-read.csv("Test_mod_03_preds.csv", head=T, sep=",")
SPDATA_test_03<-SPDATA_test_03[,2:8]
SPDATA_test_04<-read.csv("Test_mod_04_preds.csv", head=T, sep=",")
SPDATA_test_04<-SPDATA_test_04[,2:8]
SPDATA_test_05<-read.csv("Test_mod_05_preds.csv", head=T, sep=",")
SPDATA_test_05<-SPDATA_test_05[,2:8]
SPDATA_test_06<-read.csv("Test_mod_06_preds.csv", head=T, sep=",")
SPDATA_test_06<-SPDATA_test_06[,2:8]
SPDATA_test_07<-read.csv("Test_mod_07_preds.csv", head=T, sep=",")
SPDATA_test_07<-SPDATA_test_07[,2:8]
SPDATA_test_08<-read.csv("Test_mod_08_preds.csv", head=T, sep=",")
SPDATA_test_08<-SPDATA_test_08[,2:8]
SPDATA_test_09<-read.csv("Test_mod_09_preds.csv", head=T, sep=",")
SPDATA_test_09<-SPDATA_test_09[,2:8]
SPDATA_test_10<-read.csv("Test_mod_10_preds.csv", head=T, sep=",")
SPDATA_test_10<-SPDATA_test_10[,2:8]

SPDATA_train_01<-read.csv("Training_mod_01_preds.csv", head=T, sep=",")
SPDATA_train_01<-SPDATA_train_01[,2:8]
SPDATA_train_02<-read.csv("Training_mod_02_preds.csv", head=T, sep=",")
SPDATA_train_02<-SPDATA_train_02[,2:8]
SPDATA_train_03<-read.csv("Training_mod_03_preds.csv", head=T, sep=",")
SPDATA_train_03<-SPDATA_train_03[,2:8]
SPDATA_train_04<-read.csv("Training_mod_04_preds.csv", head=T, sep=",")
SPDATA_train_04<-SPDATA_train_04[,2:8]
SPDATA_train_05<-read.csv("Training_mod_05_preds.csv", head=T, sep=",")
SPDATA_train_05<-SPDATA_train_05[,2:8]
SPDATA_train_06<-read.csv("Training_mod_06_preds.csv", head=T, sep=",")
SPDATA_train_06<-SPDATA_train_06[,2:8]
SPDATA_train_07<-read.csv("Training_mod_07_preds.csv", head=T, sep=",")
SPDATA_train_07<-SPDATA_train_07[,2:8]
SPDATA_train_08<-read.csv("Training_mod_08_preds.csv", head=T, sep=",")
SPDATA_train_08<-SPDATA_train_08[,2:8]
SPDATA_train_09<-read.csv("Training_mod_09_preds.csv", head=T, sep=",")
SPDATA_train_09<-SPDATA_train_09[,2:8]
SPDATA_train_10<-read.csv("Training_mod_10_preds.csv", head=T, sep=",")
SPDATA_train_10<-SPDATA_train_10[,2:8]

SPDATA_full<-read.csv("Full_mod_preds.csv", head=T, sep=",")
SPDATA_full<-SPDATA_full[,2:8]

# Getting values for TEST
# Create a df to put results into - it has 13 rows and 9 columns
df_res_test <- data.frame(Test01 = double(length = 13),
                          Test02 = double(length = 13),
                          Test03 = double(length = 13),
                          Test04 = double(length = 13),
                          Test05 = double(length = 13),
                          Test06 = double(length = 13),
                          Test07 = double(length = 13),
                          Test08 = double(length = 13),
                          Test09 = double(length = 13),
                          Test10 = double(length = 13))
```

```
rownames(df_res_test) <- c("AUC",
        "Default",
        "Sens=Spec",
        "MaxSens+Spec",
        "MaxKappa",
        "MaxPCC",
        "PredPrev=Obs",
        "ObsPrev",
        "MeanProb",
        "MinROCdist",
        "ReqSens",
        "ReqSpect",
        "Cost")

#calculate AUC
# Find optimal thresholds, use to decide on a few thresholds to apply to find out how the accuracy statistics are behaving

df_res_test[1,1] = auc(SPDATA_test_01,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_01,which.model=1)
df_res_test[2:13,1]=opti_thres$Cloglog

df_res_test[1,2] = auc(SPDATA_test_02,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_02,which.model=1)
df_res_test[2:13,2]=opti_thres$Cloglog

df_res_test[1,3] = auc(SPDATA_test_03,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_03,which.model=1)
df_res_test[2:13,3]=opti_thres$Cloglog

df_res_test[1,4] = auc(SPDATA_test_04,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_04,which.model=1)
df_res_test[2:13,4]=opti_thres$Cloglog

df_res_test[1,5] = auc(SPDATA_test_05,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_05,which.model=1)
df_res_test[2:13,5]=opti_thres$Cloglog

df_res_test[1,6] = auc(SPDATA_test_06,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_06,which.model=1)
df_res_test[2:13,6]=opti_thres$Cloglog

df_res_test[1,7] = auc(SPDATA_test_07,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_07,which.model=1)
df_res_test[2:13,7]=opti_thres$Cloglog

df_res_test[1,8] = auc(SPDATA_test_08,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_08,which.model=1)
df_res_test[2:13,8]=opti_thres$Cloglog

df_res_test[1,9] = auc(SPDATA_test_09,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_09,which.model=1)
df_res_test[2:13,9]=opti_thres$Cloglog

df_res_test[1,10] = auc(SPDATA_test_10,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_test_10,which.model=1)
df_res_test[2:13,10]=opti_thres$Cloglog

Test_AUC_mean = rowMeans(df_res_test)[1]
```

```
mat_res_test = as.matrix(df_res_test)
Test_AUC_std = rowSds(mat_res_test)[1]

# Getting values for TRAINING
# Create a df to put results into - it has 13 rows and 9 columns
df_res_train <- data.frame(Train01 = double(length = 13),
                           Train02 = double(length = 13),
                           Train03 = double(length = 13),
                           Train04 = double(length = 13),
                           Train05 = double(length = 13),
                           Train06 = double(length = 13),
                           Train07 = double(length = 13),
                           Train08 = double(length = 13),
                           Train09 = double(length = 13),
                           Train10 = double(length = 13))

rownames(df_res_train) <- c("AUC",
                           "Default",
                           "Sens=Spec",
                           "MaxSens+Spec",
                           "MaxKappa",
                           "MaxPCC",
                           "PredPrev=Obs",
                           "ObsPrev",
                           "MeanProb",
                           "MinROCdist",
                           "ReqSens",
                           "ReqSpect",
                           "Cost")

#calculate AUC
#find optimal thresholds, use to decide on a few decent thresholds to apply to find out how the accuracy statistics are
behaving - copy and paste out

df_res_train[1,1] = auc(SPDATA_train_01,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_01,which.model=1)
df_res_train[2:13,1]=opti_thres$Cloglog

df_res_train[1,2] = auc(SPDATA_train_02,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_02,which.model=1)
df_res_train[2:13,2]=opti_thres$Cloglog

df_res_train[1,3] = auc(SPDATA_train_03,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_03,which.model=1)
df_res_train[2:13,3]=opti_thres$Cloglog

df_res_train[1,4] = auc(SPDATA_train_04,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_04,which.model=1)
df_res_train[2:13,4]=opti_thres$Cloglog

df_res_train[1,5] = auc(SPDATA_train_05,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_05,which.model=1)
df_res_train[2:13,5]=opti_thres$Cloglog

df_res_train[1,6] = auc(SPDATA_train_06,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_06,which.model=1)
df_res_train[2:13,6]=opti_thres$Cloglog

df_res_train[1,7] = auc(SPDATA_train_07,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_07,which.model=1)
```

```
df_res_train[2:13,7]=opti_thres$Cloglog

df_res_train[1,8] = auc(SPDATA_train_08,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_08,which.model=1)
df_res_train[2:13,8]=opti_thres$Cloglog

df_res_train[1,9] = auc(SPDATA_train_09,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_09,which.model=1)
df_res_train[2:13,9]=opti_thres$Cloglog

df_res_train[1,10] = auc(SPDATA_train_10,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_train_10,which.model=1)
df_res_train[2:13,10]=opti_thres$Cloglog


Train_AUC_mean = rowMeans(df_res_train)[1]
mat_res_train = as.matrix(df_res_train)
Train_AUC_std = rowSds(mat_res_train)[1]
```

```
# Getting values for the FULL MODEL
# Create a df to put results into - it has 13 rows and 9 columns
df_res_full <- data.frame(FullModel = double(length = 13))
rownames(df_res_full) <- c("AUC",
                           "Default",
                           "Sens=Spec",
                           "MaxSens+Spec",
                           "MaxKappa",
                           "MaxPCC",
                           "PredPrev=Obs",
                           "ObsPrev",
                           "MeanProb",
                           "MinROCdist",
                           "ReqSens",
                           "ReqSpect",
                           "Cost")

# Calculate AUC
df_res_full[1,1] = auc(SPDATA_full,st.dev=TRUE,which.model=1)
opti_thres = optimal.thresholds(SPDATA_full,which.model=1)
df_res_full[2:13,1]=opti_thres$Cloglog

Full_AUC_mean =df_res_full[1,1]
Full_AUC_std = 0

# plot the 3 means and their standard deviation
df_auc <- data.frame(
                  Mean= double(length = 3),
                   Std = double(length = 3))

rownames(df_auc) <- c("Full",
                      "Train",
                      "Test")

df_auc$Model = character(length = 3)

df_auc$Model = c("Full",
                 "Train",
                 "Test")

df_auc[1,1]= Full_AUC_mean
df_auc[1,2]= Full_AUC_std
df_auc[2,1]= Train_AUC_mean
df_auc[2,2]= Train_AUC_std
df_auc[3,1]= Test_AUC_mean
df_auc[3,2]= Test_AUC_std

print("Mean AUC and standard deviation for the full, train and test models")
df_auc

# quick dirty graph to see how they compare
ggplot(df_auc, aes(x=Model, y=Mean)) +
  geom_errorbar(aes(ymin=Mean-Std, ymax=Mean+Std), width=.2) +
  geom_line() +
  geom_point()
#results given as csv
write.csv(df_auc,"RES_Model_AUC_comparison.csv")
write.csv(df_res_test,"RES_TEST_model_thresholding_methods.csv")
write.csv(df_res_train,"RES_TRAIN_model_thresholding_methods.csv")
```

```
write.csv(df_res_full,"RES_FULL_model_thresholding_methods.csv")

#### --- STEP 4 --- ####
# Selection of the thresholding methods, we agreed that the best thresholdind methods to keep were:
#- MaxKappa to have integrated measure of model preformance -> row 5
#- MaxSens+Spe to have the true skill of model -> row 4
#- PredPrev=obs to assess prevalence, an important parameter for our model -> row 7

# Accuracy shows you the PPC senstitivity and specificity for your elected thresholds with their associated standard deviations.
# So we get the performances of our model for each train, test and full datasets when using the 3 thresholds given by the selected thresholding method MaxKappa,  MaxSens+Spe, PredPrev=obs

### TEST DATASET
paa_test_01<-presence.absence.accuracy(SPDATA_test_01,threshold=c(df_res_test[4,1],df_res_test[5,1],     df_res_test[7,1]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_01<-paa_test_01[,2:ncol(paa_test_01)]
rownames(paa_test_01) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_02<-presence.absence.accuracy(SPDATA_test_02,threshold=c(df_res_test[4,2],df_res_test[5,2],     df_res_test[7,2]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_02<-paa_test_02[,2:ncol(paa_test_02)]
rownames(paa_test_02) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_03<-presence.absence.accuracy(SPDATA_test_03,threshold=c(df_res_test[4,3],df_res_test[5,3],     df_res_test[7,3]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_03<-paa_test_03[,2:ncol(paa_test_03)]
rownames(paa_test_03) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_04<-presence.absence.accuracy(SPDATA_test_04,threshold=c(df_res_test[4,4],df_res_test[5,4],     df_res_test[7,4]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_04<-paa_test_04[,2:ncol(paa_test_04)]
rownames(paa_test_04) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
```

```
paa_test_05<-presence.absence.accuracy(SPDATA_test_05,threshold=c(df_res_test[4,5],df_res_test[5,5], df_res_
test[7,5]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_05<-paa_test_05[,2:ncol(paa_test_05)]
rownames(paa_test_05) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_06<-presence.absence.accuracy(SPDATA_test_06,threshold=c(df_res_test[4,6],df_res_test[5,6], df_res_
test[7,6]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_06<-paa_test_06[,2:ncol(paa_test_06)]
rownames(paa_test_06) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_07<-presence.absence.accuracy(SPDATA_test_07,threshold=c(df_res_test[4,7],df_res_test[5,7], df_res_
test[7,7]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_07<-paa_test_07[,2:ncol(paa_test_07)]
rownames(paa_test_07) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_08<-presence.absence.accuracy(SPDATA_test_08,threshold=c(df_res_test[4,8],df_res_test[5,8], df_res_
test[7,8]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_08<-paa_test_08[,2:ncol(paa_test_08)]
rownames(paa_test_08) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_09<-presence.absence.accuracy(SPDATA_test_09,threshold=c(df_res_test[4,9],df_res_test[5,9], df_res_
test[7,9]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_09<-paa_test_09[,2:ncol(paa_test_09)]
rownames(paa_test_09) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
paa_test_10<-presence.absence.accuracy(SPDATA_test_10,threshold=c(df_res_test[4,10],df_res_test[5,10], df_res_
test[7,10]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_test_10<-paa_test_10[,2:ncol(paa_test_10)]
rownames(paa_test_10) <- c("MaxSens+SPe",
                           "MaxKappa",
                           "PredPrev=obs")
```

```
#create a dataframe for each of the three thresholding method
Test_MaxSens_Spe=as.data.frame(rbind(paa_test_01[1,],
                                     paa_test_02[1,],
                                     paa_test_03[1,],
                                     paa_test_04[1,],
                                     paa_test_05[1,],
                                     paa_test_06[1,],
                                     paa_test_07[1,],
                                     paa_test_08[1,],
                                     paa_test_09[1,],
                                     paa_test_10[1,]))

rownames(Test_MaxSens_Spe) <- c("Test01",
                                "Test02",
                                "Test03",
                                "Test04",
                                "Test05",
                                "Test06",
                                "Test07",
                                "Test08",
                                "Test09",
                                "Test10")

Test_MaxSens_Spe = Test_MaxSens_Spe[,2:ncol(Test_MaxSens_Spe)]
mean_Test_MaxSens_Spe =  colMeans(Test_MaxSens_Spe)
mean_Test_MaxSens_Spe_value = mean_Test_MaxSens_Spe[1:5]
mean_Test_MaxSens_Spe_std = mean_Test_MaxSens_Spe[6:10]

write.csv(Test_MaxSens_Spe,"RES_detailed_Test_MaxSens_Spe.csv")

Test_MaxKappa=as.data.frame(rbind(paa_test_01[2,],
                                  paa_test_02[2,],
                                  paa_test_03[2,],
                                  paa_test_04[2,],
                                  paa_test_05[2,],
                                  paa_test_06[2,],
                                  paa_test_07[2,],
                                  paa_test_08[2,],
                                  paa_test_09[2,],
                                  paa_test_10[2,]))

rownames(Test_MaxKappa) <- c("Test01",
                             "Test02",
                             "Test03",
                             "Test04",
                             "Test05",
                             "Test06",
                             "Test07",
                             "Test08",
                             "Test09",
                             "Test10")

Test_MaxKappa = Test_MaxKappa[,2:ncol(Test_MaxKappa)]
mean_Test_MaxKappa =  colMeans(Test_MaxKappa)
mean_Test_MaxKappa_value = mean_Test_MaxKappa[1:5]
mean_Test_MaxKappa_std = mean_Test_MaxKappa[6:10]
write.csv(Test_MaxKappa,"RES_detailed_Test_MaxKappa.csv")

Test_PredPrev_obs= as.data.frame(rbind(paa_test_01[3,],
```

```
                        paa_test_02[3,],
                        paa_test_03[3,],
                        paa_test_04[3,],
                        paa_test_05[3,],
                        paa_test_06[3,],
                        paa_test_07[3,],
                        paa_test_08[3,],
                        paa_test_09[3,],
                        paa_test_10[3,]))

rownames(Test_PredPrev_obs) <- c("Test01",
                        "Test02",
                        "Test03",
                        "Test04",
                        "Test05",
                        "Test06",
                        "Test07",
                        "Test08",
                        "Test09",
                        "Test10")

Test_PredPrev_obs = Test_PredPrev_obs[,2:ncol(Test_PredPrev_obs)]
mean_Test_PredPrev_obs =  colMeans(Test_PredPrev_obs)
mean_Test_PredPrev_obs_value = mean_Test_PredPrev_obs[1:5]
mean_Test_PredPrev_obs_std = mean_Test_PredPrev_obs[6:10]

write.csv(Test_PredPrev_obs,"RES_detailed_Test_PredPrev_obs.csv")

#create a dataframe with the average value for Test for each of the three method  for parameter value

Mean_Test_value  =  t(rbind(mean_Test_MaxSens_Spe_value,mean_Test_MaxKappa_value,mean_Test_PredPrev_
obs_value))
Mean_Test_value=as.data.frame(Mean_Test_value)
Mean_of_mean_test_value = colMeans(Mean_Test_value)
Mean_Test_value = rbind(Mean_Test_value, Mean_of_mean_test_value)
rownames(Mean_Test_value) <- c("PCC",
                        "sensitivity",
                        "specificity",
                        "Kappa",
                        "AUC",
                        "Average")
Mean_Test_value$Model = character(length = 6)
Mean_Test_value[Mean_Test_value==""]<-"Test"

write.csv(Mean_Test_value,"RES_Test_mean_value.csv")

#create a dataframe with the average value for Test for each of the three method  for parameter std
Mean_Test_std = t(rbind(mean_Test_MaxSens_Spe_std,mean_Test_MaxKappa_std,mean_Test_PredPrev_obs_std))
Mean_Test_std=as.data.frame(Mean_Test_std)
Mean_of_mean_test_std = colMeans(Mean_Test_std)
Mean_Test_std = rbind(Mean_Test_std, Mean_of_mean_test_std)
rownames(Mean_Test_std) <- c("PCC.sd",
                        "sensitivity.sd",
                        "specificity.sd",
                        "Kappa.sd",
                        "AUC.sd",
                        "Average")
Mean_Test_std$Model = character(length = 6)
Mean_Test_std[Mean_Test_std==""]<-"Test"
```

```
write.csv(Mean_Test_value,"RES_TEST_std_mean_value.csv")

### TRAIN DATASET
paa_train_01<-presence.absence.accuracy(SPDATA_train_01,threshold=c(df_res_train[4,1],df_res_train[5,1],  df_
res_train[7,1]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_01<-paa_train_01[,2:ncol(paa_train_01)]
rownames(paa_train_01) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_02<-presence.absence.accuracy(SPDATA_train_02,threshold=c(df_res_train[4,2],df_res_train[5,2],  df_
res_train[7,2]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_02<-paa_train_02[,2:ncol(paa_train_02)]
rownames(paa_train_02) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_03<-presence.absence.accuracy(SPDATA_train_03,threshold=c(df_res_train[4,3],df_res_train[5,3],  df_
res_train[7,3]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_03<-paa_train_03[,2:ncol(paa_train_03)]
rownames(paa_train_03) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_04<-presence.absence.accuracy(SPDATA_train_04,threshold=c(df_res_train[4,4],df_res_train[5,4],  df_
res_train[7,4]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_04<-paa_train_04[,2:ncol(paa_train_04)]
rownames(paa_train_04) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_05<-presence.absence.accuracy(SPDATA_train_05,threshold=c(df_res_train[4,5],df_res_train[5,5],  df_
res_train[7,5]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_05<-paa_train_05[,2:ncol(paa_train_05)]
rownames(paa_train_05) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_06<-presence.absence.accuracy(SPDATA_train_06,threshold=c(df_res_train[4,6],df_res_train[5,6],  df_
res_train[7,6]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_06<-paa_train_06[,2:ncol(paa_train_06)]
rownames(paa_train_06) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_07<-presence.absence.accuracy(SPDATA_train_07,threshold=c(df_res_train[4,7],df_res_train[5,7],  df_
res_train[7,7]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_07<-paa_train_07[,2:ncol(paa_train_07)]
rownames(paa_train_07) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_08<-presence.absence.accuracy(SPDATA_train_08,threshold=c(df_res_train[4,8],df_res_train[5,8],  df_
res_train[7,8]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_08<-paa_train_08[,2:ncol(paa_train_08)]
rownames(paa_train_08) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_09<-presence.absence.accuracy(SPDATA_train_09,threshold=c(df_res_train[4,9],df_res_train[5,9],  df_
res_train[7,9]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_09<-paa_train_09[,2:ncol(paa_train_09)]
rownames(paa_train_09) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")
paa_train_10<-presence.absence.accuracy(SPDATA_train_10,threshold=c(df_res_train[4,10],df_res_train[5,10], df_
res_train[7,10]), find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_train_10<-paa_train_10[,2:ncol(paa_train_10)]
```

```
rownames(paa_train_10) <- c("MaxSens+SPe",
                            "MaxKappa",
                            "PredPrev=obs")

#create a dataframe for each of the three thresholding method
Train_MaxSens_Spe=as.data.frame(rbind(paa_train_01[1,],
                                      paa_train_02[1,],
                                      paa_train_03[1,],
                                      paa_train_04[1,],
                                      paa_train_05[1,],
                                      paa_train_06[1,],
                                      paa_train_07[1,],
                                      paa_train_08[1,],
                                      paa_train_09[1,],
                                      paa_train_10[1,]))




rownames(Train_MaxSens_Spe) <- c("Train01",
                                 "Train02",
                                 "Train03",
                                 "Train04",
                                 "Train05",
                                 "Train06",
                                 "Train07",
                                 "Train08",
                                 "Train09",
                                 "Train10")

Train_MaxSens_Spe = Train_MaxSens_Spe[,2:ncol(Train_MaxSens_Spe)]
mean_Train_MaxSens_Spe =  colMeans(Train_MaxSens_Spe)
mean_Train_MaxSens_Spe_value = mean_Train_MaxSens_Spe[1:5]
mean_Train_MaxSens_Spe_std = mean_Train_MaxSens_Spe[6:10]

write.csv(Train_MaxSens_Spe,"RES_detailed_Train_MaxSens_Spe.csv")

Train_MaxKappa=as.data.frame(rbind(paa_train_01[2,],
                                   paa_train_02[2,],
                                   paa_train_03[2,],
                                   paa_train_04[2,],
                                   paa_train_05[2,],
                                   paa_train_06[2,],
                                   paa_train_07[2,],
                                   paa_train_08[2,],
                                   paa_train_09[2,],
                                   paa_train_10[2,]))

rownames(Train_MaxKappa) <- c("Train01",
                              "Train02",
                              "Train03",
                              "Train04",
                              "Train05",
                              "Train06",
                              "Train07",
                              "Train08",
                              "Train09",
                              "Train10")

Train_MaxKappa = Train_MaxKappa[,2:ncol(Train_MaxKappa)]
```

```
mean_Train_MaxKappa =  colMeans(Train_MaxKappa)
mean_Train_MaxKappa_value = mean_Train_MaxKappa[1:5]
mean_Train_MaxKappa_std = mean_Train_MaxKappa[6:10]

write.csv(Train_MaxKappa,"RES_detailed_Train_MaxKappa.csv")

Train_PredPrev_obs= as.data.frame(rbind(paa_train_01[3,],
                                        paa_train_02[3,],
                                        paa_train_03[3,],
                                        paa_train_04[3,],
                                        paa_train_05[3,],
                                        paa_train_06[3,],
                                        paa_train_07[3,],
                                        paa_train_08[3,],
                                        paa_train_09[3,],
                                        paa_train_10[3,]))

rownames(Train_PredPrev_obs) <- c("Train01",
                                  "Train02",
                                  "Train03",
                                  "Train04",
                                  "Train05",
                                  "Train06",
                                  "Train07",
                                  "Train08",
                                  "Train09",
                                  "Train10")

Train_PredPrev_obs = Train_PredPrev_obs[,2:ncol(Train_PredPrev_obs)]
mean_Train_PredPrev_obs =  colMeans(Train_PredPrev_obs)
mean_Train_PredPrev_obs_value = mean_Train_PredPrev_obs[1:5]
mean_Train_PredPrev_obs_std = mean_Train_PredPrev_obs[6:10]

write.csv(Train_PredPrev_obs,"RES_TRAIN_detailed_PredPrev_obs.csv")

# Create a dataframe with the average value for Train for each of the three method  for parameter value
Mean_Train_value  =  t(rbind(mean_Train_MaxSens_Spe_value,mean_Train_MaxKappa_value,mean_Train_Pred-
Prev_obs_value))
Mean_Train_value=as.data.frame(Mean_Train_value)
Mean_of_mean_Train_value = colMeans(Mean_Train_value)
Mean_Train_value = rbind(Mean_Train_value, Mean_of_mean_Train_value)

rownames(Mean_Train_value) <- c("PCC",
                                "sensitivity",
                                "specificity",
                                "Kappa",
                                "AUC",
                                "Average")

Mean_Train_value$Model = character(length = 6)
Mean_Train_value[Mean_Train_value==""]<-"Train"

write.csv(Mean_Train_value,"RES_Train_mean_value.csv")

#create a dataframe with the average value for Train for each of the three method  for parameter std
Mean_Train_std = t(rbind(mean_Train_MaxSens_Spe_std,mean_Train_MaxKappa_std,mean_Train_PredPrev_obs_
std))
Mean_Train_std=as.data.frame(Mean_Train_std)
Mean_of_mean_Train_std = colMeans(Mean_Train_std)
Mean_Train_std = rbind(Mean_Train_std, Mean_of_mean_Train_std)
```

```
rownames(Mean_Train_std) <- c("PCC.sd",
                        "sensitivity.sd",
                        "specificity.sd",
                        "Kappa.sd",
                        "AUC.sd",
                        "Average")

Mean_Train_std$Model = character(length = 6)
Mean_Train_std[Mean_Train_std==""]<-"Train"

write.csv(Mean_Train_std,"RES_Train_mean_std.csv")

### FULL DATASET
paa_full<-presence.absence.accuracy(SPDATA_full,threshold=c(df_res_full[4,1],df_res_full[5,1],  df_res_full[7,1]),
find.auc=TRUE, st.dev=TRUE, which.model=1)
paa_full<-paa_full[,2:ncol(paa_full)]
rownames(paa_full) <- c("MaxSens+SPe",
                        "MaxKappa",
                        "PredPrev=obs")

write.csv(paa_full,"RES_detailed_full.csv")

mat_paa_full = t(as.matrix(paa_full))
paa_full_t = as.data.frame(mat_paa_full)
paa_full_value =paa_full_t[2:6,]
Mean_of_paa_full_value = colMeans(paa_full_value)
Mean_paa_full_value = rbind(paa_full_value, Mean_of_paa_full_value)
rownames(Mean_paa_full_value) <- c("PCC",
                              "sensitivity",
                              "specificity",
                              "Kappa",
                              "AUC",
                              "Average")
Mean_paa_full_value$Model = character(length = 6)
Mean_paa_full_value[Mean_paa_full_value==""]<-"Full"

write.csv(Mean_paa_full_value,"RES_Full_mean_value.csv")

paa_full_std =paa_full_t[7:11,]
Mean_of_paa_full_std = colMeans(paa_full_std)
Mean_paa_full_std = rbind(paa_full_std, Mean_of_paa_full_std)
rownames(Mean_paa_full_std) <- c("PCC",
                              "sensitivity",
                              "specificity",
                              "Kappa",
                              "AUC",
                              "Average")
Mean_paa_full_std$Model = character(length = 6)
Mean_paa_full_std[Mean_paa_full_std==""]<-"Full"

write.csv(Mean_paa_full_std,"RES_Full_mean_std.csv")


#### --- STEP 5 --- ####
# Putting the final results into one csv

#giving same column name to all dataframes
names(Mean_Test_value)[1]<- 'MaxSens_Spec_VALUE'
names(Mean_Test_value)[2]<- 'MaxKappa_VALUE'
names(Mean_Test_value)[3]<- 'PredPrev_obs_VALUE'
```

```
names(Mean_Train_value)[1]<- ‘MaxSens_Spec_VALUE’
names(Mean_Train_value)[2]<- ‘MaxKappa_VALUE’
names(Mean_Train_value)[3]<- ‘PredPrev_obs_VALUE’

names(Mean_paa_full_value)[1]<- ‘MaxSens_Spec_VALUE’
names(Mean_paa_full_value)[2]<- ‘MaxKappa_VALUE’
names(Mean_paa_full_value)[3]<- ‘PredPrev_obs_VALUE’


names(Mean_Test_std)[1]<- ‘MaxSens_Spec_STD’
names(Mean_Test_std)[2]<- ‘MaxKappa_STD’
names(Mean_Test_std)[3]<- ‘PredPrev_obs_STD’

names(Mean_Train_std)[1]<- ‘MaxSens_Spec_STD’
names(Mean_Train_std)[2]<- ‘MaxKappa_STD’
names(Mean_Train_std)[3]<- ‘PredPrev_obs_STD’

names(Mean_paa_full_std)[1]<- ‘MaxSens_Spec_STD’
names(Mean_paa_full_std)[2]<- ‘MaxKappa_STD’
names(Mean_paa_full_std)[3]<- ‘PredPrev_obs_STD’

threshold_full=c(df_res_full[4,1],df_res_full[5,1],     df_res_full[7,1],””,df_res_full[4,1],df_res_full[5,1],     df_res_
full[7,1],””)

Value_all = rbind(Mean_Test_value,Mean_Train_value,Mean_paa_full_value)
std_all = rbind(Mean_Test_std,Mean_Train_std,Mean_paa_full_std)

final_all_res = cbind(Value_all,std_all)
final_all_res = rbind(final_all_res,threshold_full)
rownames(final_all_res)[19]<-”Threshold”

write.csv(final_all_res,”RES_FINAL.csv”)
```