R code: Spatial cross-validation

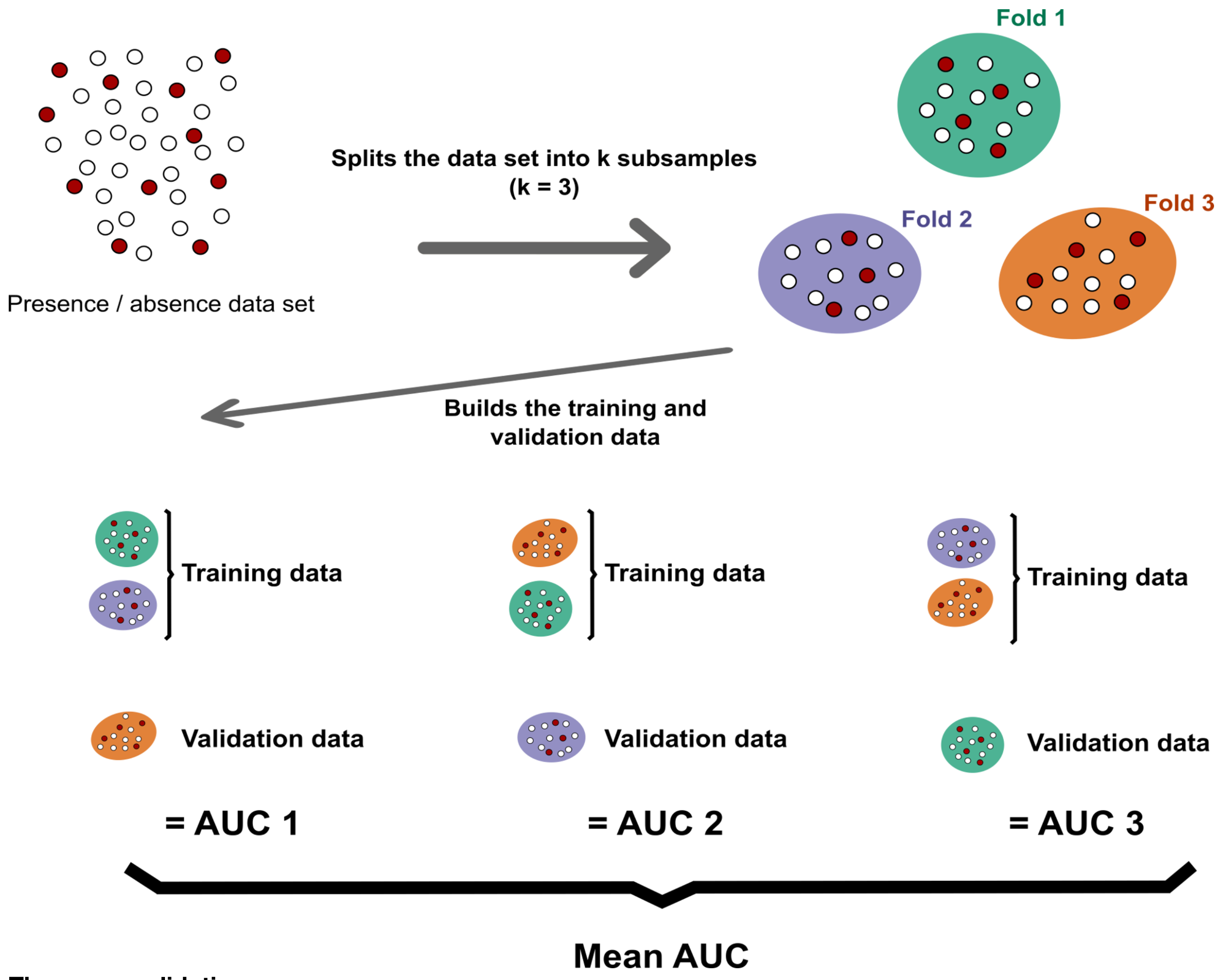
Jean Artois, Madhur Dhingra, Marius Gilbert

01 feb 2017

The following example is taking from Dhingra et al. 2016 (https://elifesciences.org/content/5/e19571)

I. Spatial cross-validation for assessing transferability of ecological model

The Boosted Regression Tree models (BRT) belongs to the family of machine learning methods. BRT can be used within the framework of Species Distribution Modeling (SDM). A common application of SDM is to predict species ranges with environmental data as predictors. However, these methods need a cross-validation (CV) which can quantify the transferability of models over independant dataset (Wenger & Olden 2012 (http://onlinelibrary.wiley.com/doi/10.1111/j.2041-210X.2011.00170.x/abstract)). E.g., cross-validation estimates the expected goodness of fit of a model to a dataset (testing-data) that is independent of the data that were used to train the model (training-data).



The cross-validation

Due to spatial auto-correlation, the training and testing-data, randomly sampled, are rarely independant. In that situation, the performance metrics of models can overestimate the degree of model fitting. The following are proposed methodologies for cross-validation of the models taking into account the spatial information during the sampling of training/testing-data.
We use the H5N1 dataset as an example to demonstrate the various methods of cross-validation. HPAI subtype H5N1 presence-data was collected from the database of the Global Animal Health Information System of the FAO (EMPRES-i (http://empres-i.fao.org/)). We aim to predict areas of habitat suitability of HPAI H5N1, in order to inform surveillance protocols and enable early detection at a global scale. For this purpose, the predictors were categorized into three sets of variables in order to test and to compare the ability of sets to predict and generalize:
- **Set 1** includes the host density variables
- **Set 2** includes Set 1 variables and all landuse/landcover (LU/LC) variables.
- **Set 3** includes Set 1 and all temperature and NDVI related eco-climatic variables.

SDM requires information on spatial presence or occurrence data, a background of pseudo-absences or at risk reas, and information on covariates that are likely to affect distribution of HPAI across the global landscape. For this study, we sampled pseudo-absences from background areas with human population density having poultry density within the area unit.

II. Distribution maps of HPAI H5N1 records

The aim of this section is to plot a distribution map with R functions for spatial data

```r
## Loads the following libraries:
# Manages GIS
library(rgdal)
library(raster)

# Fits and evaluates BRT
library(dismo)
library(gbm)
library(pROC)

# Finds some nice colour palettes
library(RColorBrewer)

# Loads strings as "character" format
options(stringsAsFactors = FALSE)

# Sets the general path
setwd("/home/jeanjean/Dropbox/Lubies-epi/2_Codes/Sessions/3_SDM_6_SpatialValidation")
# rmarkdown::render("SpatialValidation_V3.r", "html_document")

# Loads shapefiles
# Downloading from Natural Earth website (http://www.naturalearthdata.com/)
MyLand <- readOGR("../../Data/VNM-H5N1/2_Vector", layer = "ne_110m_land")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "../../Data/VNM-H5N1/2_Vector", layer: "ne_110m_land"
## with 127 features
## It has 2 fields
```

```r
MyGrat <- readOGR("../../Data/VNM-H5N1/2_Vector", layer = "ne_50m_graticules_20")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "../../Data/VNM-H5N1/2_Vector", layer: "ne_50m_graticules_20"
## with 27 features
## It has 6 fields
```

```r
# Sets the coordinate reference system (CRS)
P_wgs84 <- CRS(proj4string(MyLand))

# Loads a data subset about H5N1 distribution
datN1 <- read.csv("../../Data/VNM-H5N1/3_Epi/H5N1example.csv", stringsAsFactors = F)
idSp <- c(sample(which(datN1$IsPresent == 0), 2400),
          sample(which(datN1$IsPresent == 1), 350))
datN1 <- datN1[idSp,]

# Counts the presences and pseudo-absences
table(datN1$IsPresent)
```

```
##
##    0    1
## 2400  350
```

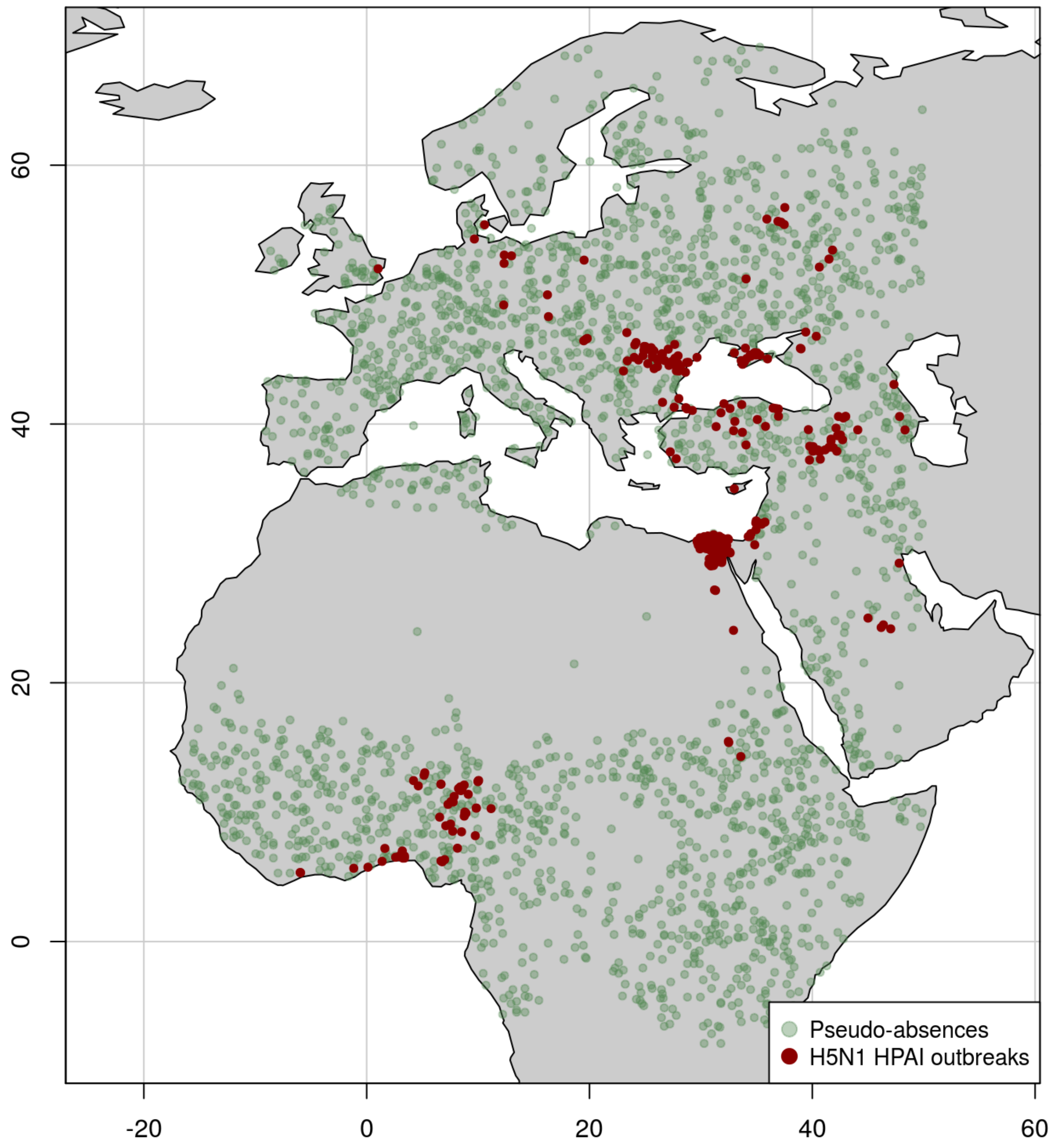
```
# Converts the data.frame into spatial object
coordinates(datN1) <- c("x", "y")
proj4string(datN1) <- P_wgs84

# Finds the spatial extent of your dataset of points
P_lim <- as.vector(extent(datN1))

# Sets the colour palette of your maps
col2rgb("palegreen4")
```

```
##       [,1]
## red     84
## green  139
## blue    84
```

```
P_col <- c(rgb(84, 139, 84, 100, max = 255), "darkred")
```

Maps the H5N1 HPAI outbreaks and pseudo-absences

```
# Plots the graticules with the right extent
plot(MyGrat, col = "grey80", xlim = P_lim[1:2], ylim = P_lim[3:4])
# Adds the axes labels
axis(side = 1, at = seq(-180, 180, 20), labels = T)
axis(side = 2, at = seq(-180, 180, 20), labels = T)
# Adds the land
plot(MyLand, add = T, col = "grey80")
# Adds your points
points(datN1, pch = 20, col = P_col[as.factor(datN1$IsPresent)])
# Draws a box around your plot
box()
# Adds the legend
legend("bottomright",
       cex = 0.9,
       legend = c("Pseudo-absences", "H5N1 HPAI outbreaks"),
       pch = 20,
       col = P_col,
       pt.cex = 2)
```



III. Data modelling

A. Standard cross-validation

Per default, the k-fold cross-validation procedure of (Elith, Leathwick & Hastie 2008) is applied inside the function `gbm.step` of ‘dismo’ package.

A.1. Model fitting

In the following script section, a BRT is fitted on each set of predictors to attempt to predict the presence of H5N1 virus.

```
# Builds a list of names. They are the predictor variables of models sorted by set.
P_VarList<-list(c("ChDnLgExt", "ChDnLgInt", "DuDnLg", "HpDnLg"),

                c("Dist_Water", "Evergreen_Deciduous", "Open_Water", "Evergreen_Broad
leaf",
                  "Deciduous_Broadleaf", "Mixed_Trees", "Shrubs", "Herbaceous_Veg",
                  "Cultivated_Managed", "Regularly_Flooded_Vegetation", "Urban_Built_
up"),

                c("Temperature_Annual_mean", "Temperature_Amplitude_annual",
                  "Temperature_Amplitude_biannual", "Temperature_Amplitude_triannual"
,
                  "Temperature_Variance_annual", "Temperature_Variance_biannual",
                  "Temperature_variance_annual_biannual_triannual", "NDVI_Annual_mea
n",
                  "NDVI_Amplitude_annual", "NDVI_Amplitude_biannual",
                  "NDVI_Amplitude_triannual", "NDVI_Variance_annual",
                  "NDVI_Variance_biannual", "NDVI_Variance_triannual",
                  "NDVI_variance_annual_biannual_triannual"))

names(P_VarList) <- list("Set1","Set2","Set3")

# Sets the run settings of BRT
P_kfold = 4 # the numbers of folds for the cross-validation
P_tc = 4 # tree complexity
P_lr = 0.01 # learning rate
P_nt = 100 # initial number of tree
P_ss = 100 # step size

# Creates an empty list to save the results of BRT
SauvBrt <- list()
datN1 <- data.frame(datN1)

# Selects the dependent variable. Finds the ID of the column "IsPresent" in datN1.
myPosDep <- match("IsPresent", names(datN1))

for(set in 1:3){
  # Selects the predictors. Finds the ID of right predictors in datN1.
  myPosPred <- match(P_VarList[[set]], names(datN1))

  # Fits a BRT
  SauvBrt[[set]] <- gbm.step(data = datN1, gbm.x = myPosPred, gbm.y = myPosDep,
                           family = "bernoulli", tree.complexity = P_tc,
                           learning.rate = P_lr, n.folds = P_kfold, n.trees = P_nt,
                           step.size= P_ss, silent = T, plot.main = F)
}

# Checks the results
SauvBrt[[1]]
```

```
## gbm::gbm(formula = y.data ~ ., distribution = as.character(family),
##     data = x.data, weights = site.weights, var.monotone = var.monotone,
##     n.trees = target.trees, interaction.depth = tree.complexity,
##     shrinkage = learning.rate, bag.fraction = bag.fraction, verbose = FALSE)
## A gradient boosted model with bernoulli loss function.
## 900 iterations were performed.
## There were 4 predictors of which 4 had non-zero influence.
```

```
summary(SauvBrt)
```

```
##      Length Class Mode
## [1,] 41     gbm   list
## [2,] 41     gbm   list
## [3,] 41     gbm   list
```

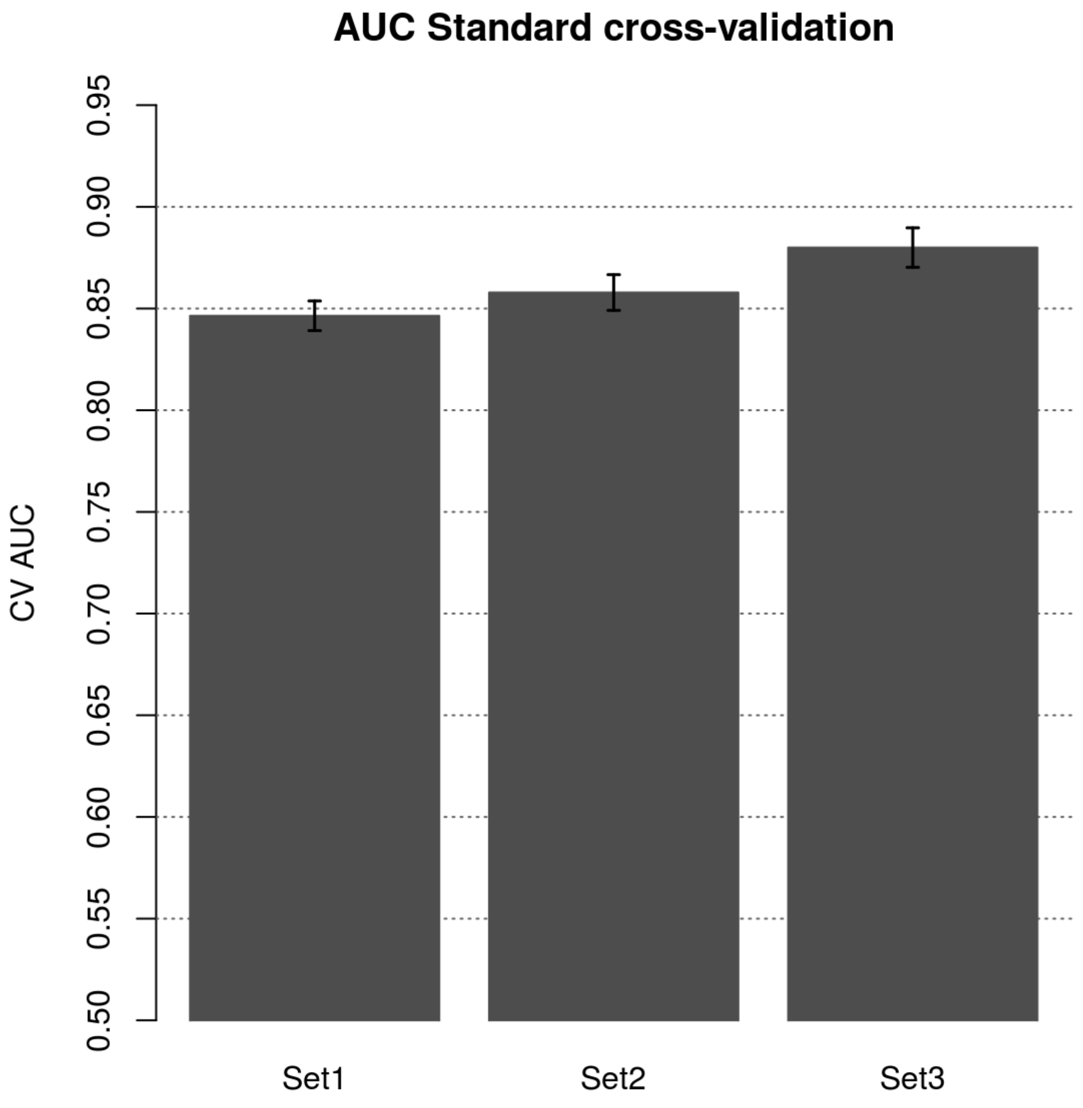
A.2. The area under the receiver operating characteristic plot

The AUC, i.e. the area under the receiver operating characteristic (ROC) plot, is a measurement of the discriminatory capacity of classification models. Taking into account sensitivity (Se), the proportion of instances of presence correctly predicted as presence, and specificity (Sp), the proportion of instances of absence correctly predicted as absence, the ROC curve plots Se versus 1-Sp across all possible thresholds. A model will be considered to discriminate better than chance if the ROC curve lies above the diagonal of no discrimination, i.e. if the AUC is higher than 0.5. This statistical summary of the ROC graph has been widely adopted by SDM researchers as a standard measure of overall discriminatory capacity because it is independent of the threshold. In other words, it avoids the potential arbitrariness associated with the selection of the threshold needed to build 2 x 2 contingency matrices. Jiménez-Valverde (2012).

```
# Some statistics of model are stocked inside the `gbm.step` output.
# For example the mean and standard error of the CV-AUC for the Set 2
# can be checked with the following indexation:
SauvBrt[[2]]$cv.statistics[c("discrimination.mean", "discrimination.se")]
```

```
## $discrimination.mean
## [1] 0.857875
##
## $discrimination.se
## [1] 0.008778513
```

```
# Extracts the AUC results from SauvBRT
Mauc <- sapply(1:3, function(ii) SauvBrt[[ii]]$cv.statistics["discrimination.mean"][[
1]])
SEauc <- sapply(1:3, function(ii) SauvBrt[[ii]]$cv.statistics["discrimination.se"][[1
]])

# Plots your results
barplot(1:3, yaxt = "n", ylim = c(0.5, 0.95), col = "white", border = "white",
        ylab = "CV AUC", names.arg = names(P_VarList),
        main = "AUC Standard cross-validation")
# Adds the y-axis labels and the background scale of the plot
axis(2, seq(0.5, 1, 0.05))
grid(col = "grey40", nx = 0, ny = 9)
# Adds the bars and saves the x coordiantes of bars
barX <- barplot(Mauc, yaxt = "n", col = "grey30", border = "grey30", add = T, xpd = F
)
# Adds the error bars
arrows(barX, Mauc - SEauc, barX,
       Mauc + SEauc, lwd = 1.5, angle = 90,
       code = 3, length = 0.03)
```



B. Spatial cross-validation

B.1. Splitting of data

In SDM, transferability has often been estimated by splitting the data set into geographically distinct subsets, fitting the model with the first subset (training-data) and validating with the second (testing-data). This is nothing more than a form of cross-validation in which the subset membership is assigned non-randomly based on a relevant factor such as geography. Wenger & Olden 2012.

To take into account the spatial information during the cross-validation of models, training and testing data are assigned non-randomly by 3 ways.

- **Distance method:** From the presence data, the `thrSample` function selects a specific number of points (equal to k, the number of folds CV) not too close of each other selected points. For this purpose, the user specifies the setting parameter "thr" corresponding to the minimum distance between the selected points. The selected points are labelled with a specific number and represent the benchmarks to build the folds CV of models. Indeed, `Lib_DistEstimatesID` identifies the nearest benchmark of each observation used in the models to assign a label at each observation (= the nearest benchmark ID). You can find the `thrSample` and `Lib_DistEstimatesID` functions at the end of this turorial (Dhingra et al., 2015).
- **Quantile method:** Corresponds to the block method of Muscarella et al., 2014 (http://onlinelibrary.wiley.com/doi/10.1111/2041-210X.12261/full) You can find the `get.block` function at the end of this turorial. The xy coordinates can be transformed (including rotation about any axis) to obtain variable outputs for this method.
- **kmeans method:** Finds k clusters in the coordinates space

In the following code section we build some maps to understand the differences between the standard cross-validation and the spatial cross-validation constructed with the 3 previous methods.

```
# This section of code partitions the dataset into a defined number of folds (P_kfol
d)

### I. The standard CV.
# Randomly samples the fraction of data to be sampled for each validation.
# The result is stocked as labels in 'LabAlea' column.
# For this example we specify a number of fold equal to 4.
# The following section of code is taken from the `gbm.step` function from dismo pack
age
presence.mask <- datN1[, "IsPresent"] == 1
absence.mask <- datN1[, "IsPresent"] == 0
n.pres <- sum(presence.mask)
n.abs <- sum(absence.mask)
datN1$LabAlea <- rep(0, nrow(datN1))
temp <- rep(seq(1, P_kfold, by = 1), length = n.pres)
temp <- temp[order(runif(n.pres, 1, 100))]
datN1$LabAlea[presence.mask] <- temp
temp <- rep(seq(1, P_kfold, by = 1), length = n.abs)
temp <- temp[order(runif(n.abs, 1, 100))]
datN1$LabAlea[absence.mask] <- temp

# A random but stratified sampling of folds
table(datN1$IsPresent, datN1$LabAlea)
```

```
##
##       1   2   3   4
##   0 600 600 600 600
##   1  88  88  87  87
```

```
### II. The spatial CV.
#### II.A. Distance method
# Selects the presence data
MyRef <- datN1[which(datN1$IsPresent==1), c("x", "y")]

# Samples the 4 benchmarks
MyId <- NULL
MyId <- thrSample(thr = 2000, coords = c("x", "y"),
                  data = MyRef, nPts = P_kfold)
MyRef <- MyRef[MyId,]

# For each point in datN1 (presence and absence), find the nearest point from "MyRef"
datN1$LabGeo1 <- Lib_DistEstimatesID(as.matrix(datN1[ ,c("x", "y")]),
                                     as.matrix(MyRef))

# Here, we have no control on the proportion of presence and pseudo-absence in each f
old
table(datN1$IsPresent, datN1$LabGeo1)
```

```
##
##       1   2   3   4
##   0 374 659 962 405
##   1 198  53  88  11
```

```
#### II.B. Quantile method
rownames(datN1) <- 1:nrow(datN1)
Occ1 <- datN1[which(datN1$IsPresent == 1), c("x", "y")]
Bg1 <- datN1[which(datN1$IsPresent == 0), c("x", "y")]

blockF <- get.block(occ = Occ1,
                    bg.coords = Bg1)

# Extracts the labels
datN1$LabGeo2 <- NA
datN1[which(datN1$IsPresent == 1), "LabGeo2"] <- blockF$occ.grp
datN1[which(datN1$IsPresent == 0), "LabGeo2"] <- blockF$bg.grp

# Here, we have only control on presence
table(datN1$IsPresent, datN1$LabGeo2)
```

```
##
##       1   2   3   4
##   0 689 392 857 462
##   1  88  87  88  87
```

```
#### II.C. Kmeans method
km_F <- kmeans(datN1[, c("x", "y")], 4, nstart = 1)
datN1$LabGeo3 <- km_F$ cluster

# Here, we have no control on the proportion of presence and pseudo-absence in each f
old
table(datN1$IsPresent, datN1$LabGeo3)
```

```
##
##        1   2   3   4
##   0 624 728 472 576
##   1  13 276  53   8
```

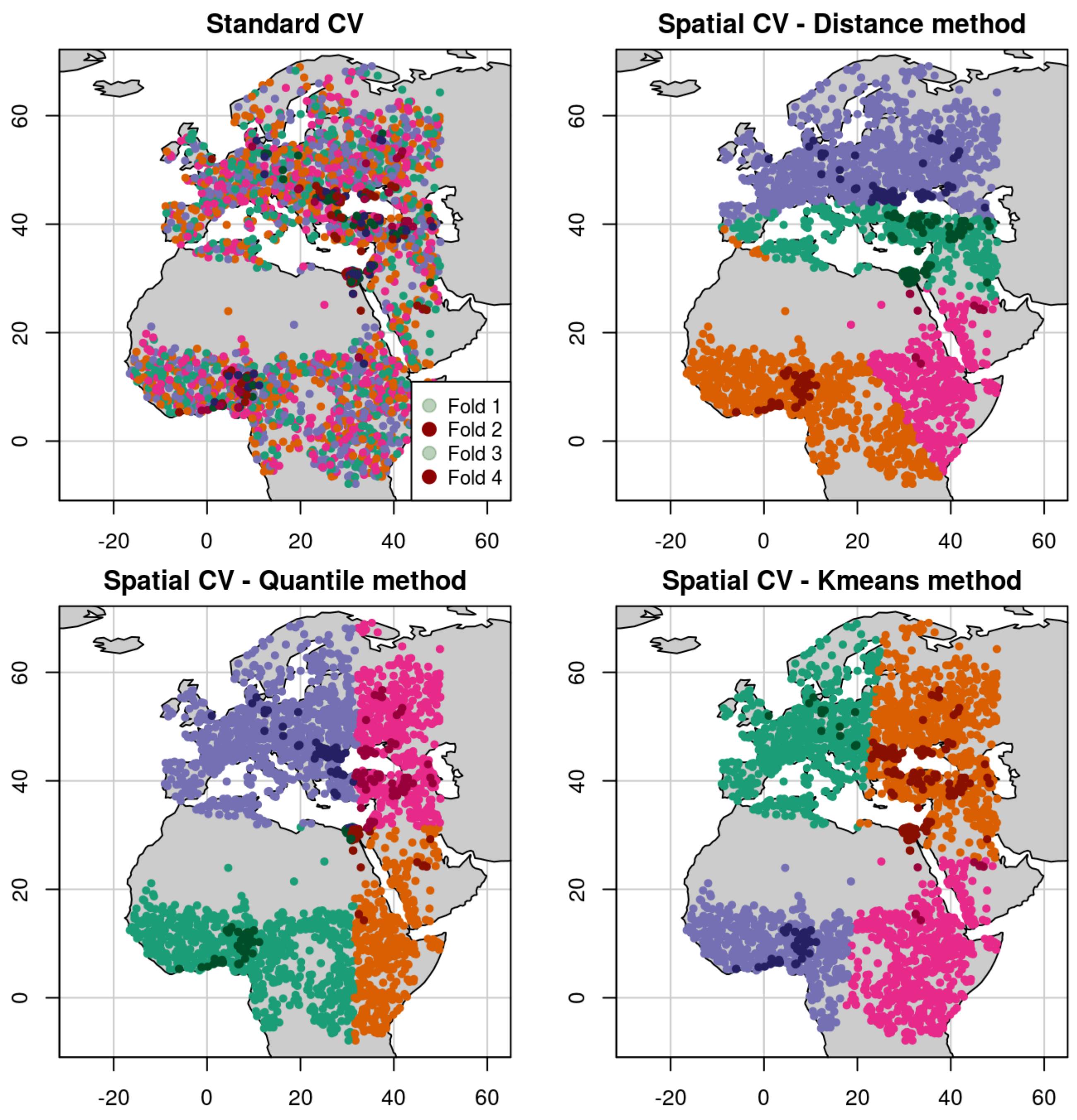
```
### III. Checks the spatial distribution of folds for each method

# Creates a colour palette with the package `RColorBrewer` for the folds
P_colA <- brewer.pal(P_kfold, "Dark2")
P_colP <- col2rgb(P_colA)
P_colP <- P_colP - 80
P_colP[which(P_colP < 0)] <- 0
P_colP <- sapply(1:4, function(ii) rgb(P_colP[1, ii], P_colP[2, ii], P_colP[3, ii], m
ax = 255))

# Codes a new factor variable to colour the map
IsP_FoldA <- as.factor(datN1$IsPresent):as.factor(datN1$LabAlea)
levels(IsP_FoldA)
```

```
## [1] "0:1" "0:2" "0:3" "0:4" "1:1" "1:2" "1:3" "1:4"
```

```
# 8 groups: presences and absences (2) * 4 folds
```

Maps the distribution of folds for each method

```
# Splits your plot window in 4 sub-windows (2 rows, 2 columns)
# and sets the margin parameters
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))

# Sandard method
plot(MyGrat, col = "grey80", xlim = P_lim[1:2], ylim = P_lim[3:4], main = "Standard C
V")
axis(side = 1, at = seq(-180, 180, 20), labels = T)
axis(side = 2, at = seq(-180, 180, 20), labels = T)
plot(MyLand, add = T, col = "grey80")
# Adds your points labeled with the folds
points(datN1[, c("x", "y")], pch = 20, col = c(P_colA, P_colP)[IsP_FoldA])
box()
legend("bottomright",
       cex = 0.9,
       legend = paste0("Fold ", 1:4),
       pch = 20,
       col = P_col,
       pt.cex = 2)

# Distance method
IsP_Fold1 <- as.factor(datN1$IsPresent):as.factor(datN1$LabGeo1)

plot(MyGrat, col = "grey80", xlim = P_lim[1:2], ylim = P_lim[3:4], main = "Spatial CV
- Distance method")
axis(side = 1, at = seq(-180, 180, 20), labels = T)
axis(side = 2, at = seq(-180, 180, 20), labels = T)
plot(MyLand, add = T, col = "grey80")
points(datN1[, c("x", "y")], pch = 20, col = c(P_colA, P_colP)[IsP_Fold1])
box()

# Quantile method
IsP_Fold2 <- as.factor(datN1$IsPresent):as.factor(datN1$LabGeo2)

plot(MyGrat, col = "grey80", xlim = P_lim[1:2], ylim = P_lim[3:4], main = "Spatial CV
- Quantile method")
axis(side = 1, at = seq(-180, 180, 20), labels = T)
axis(side = 2, at = seq(-180, 180, 20), labels = T)
plot(MyLand, add = T, col = "grey80")
points(datN1[, c("x", "y")], pch = 20, col = c(P_colA, P_colP)[IsP_Fold2])
box()

# kmeans method
IsP_Fold3 <- as.factor(datN1$IsPresent):as.factor(datN1$LabGeo3)

plot(MyGrat, col = "grey80", xlim = P_lim[1:2], ylim = P_lim[3:4], main = "Spatial CV
- Kmeans method")
axis(side = 1, at = seq(-180, 180, 20), labels = T)
axis(side = 2, at = seq(-180, 180, 20), labels = T)
plot(MyLand, add = T, col = "grey80")
# Adds your points labeled with the folds
points(datN1[, c("x", "y")], pch = 20, col = c(P_colA, P_colP)[IsP_Fold3])
box()
```



Standard CV
Spatial CV - Distance method
Spatial CV - Quantile method
Spatial CV - Kmeans method
Fold 1
Fold 2
Fold 3
Fold 4

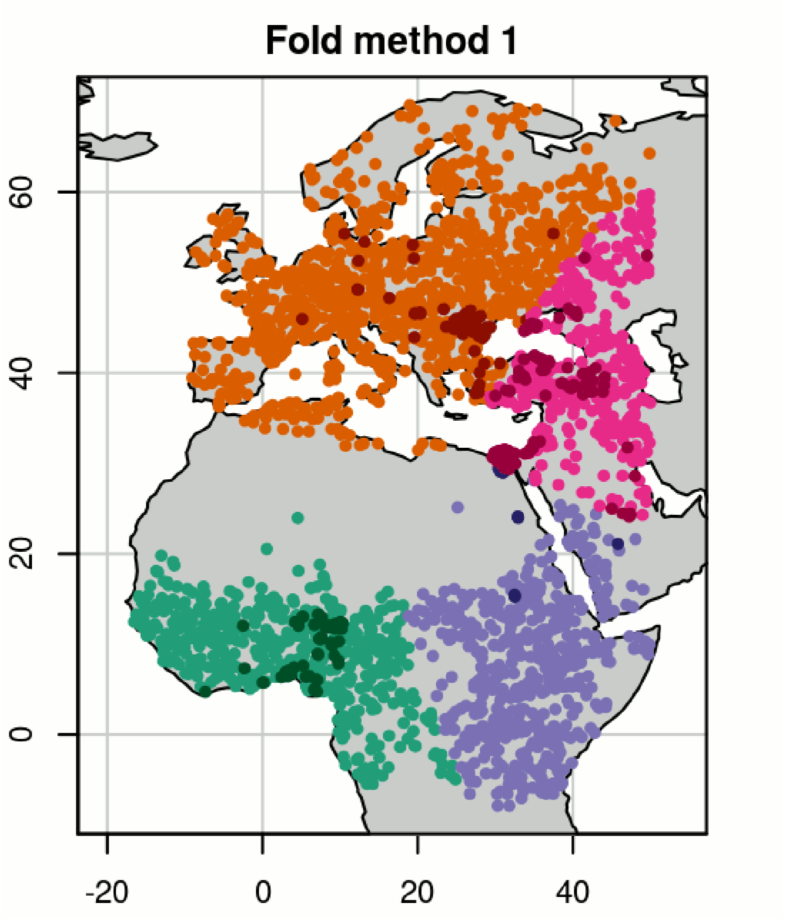
Comparison between the methods outputs:

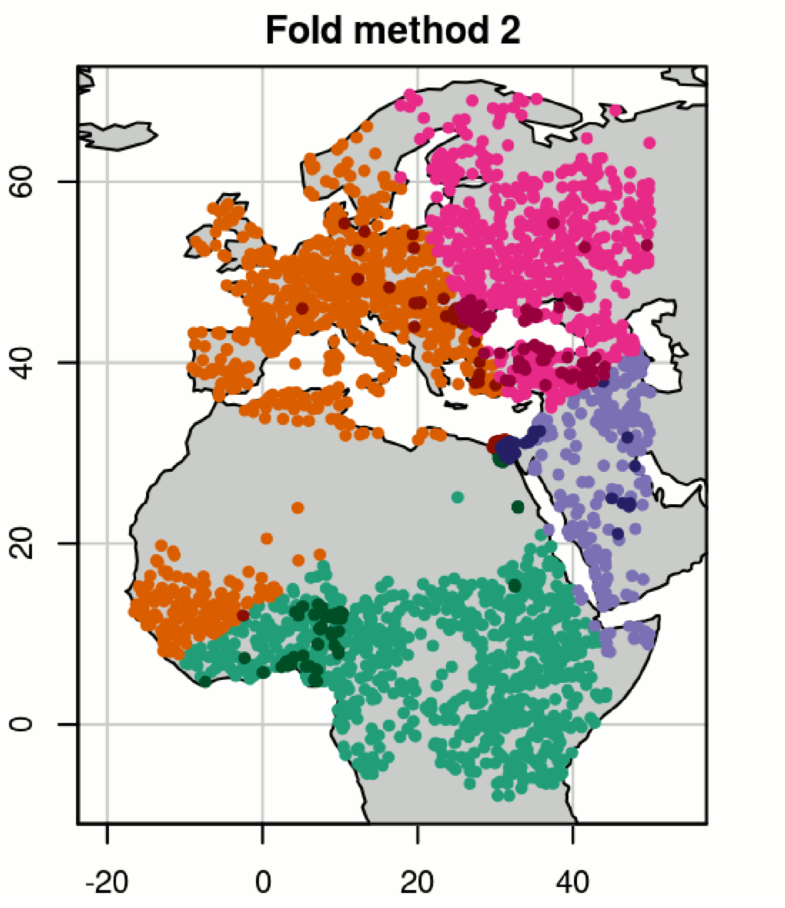
The distance method produces random outputs but the global procedure is unstable (you need to remove some set of folds which are too unbalanced in presences and absences).

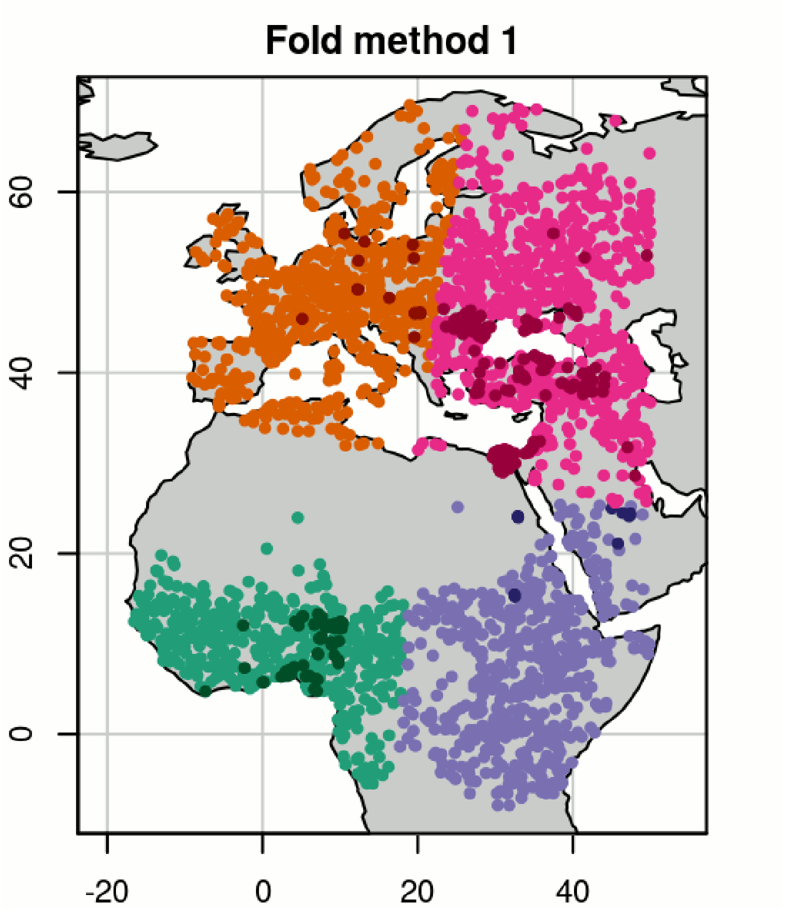
One of the main advantages to ‘Quantile method’ is that number of presence in each fold is similar (a first step to have stratify sampling). Currently, the `get.block` function is only coded for a 4-fold CV.

Method	Count 0 & 1	Resampling	Number of folds
Distance (1)	Random	Variable	Adjustable
Quantile (2)	A fixed number of 1	Variable	Fixed to 4
kmeans (3)	Random	No variation	Adjustable

Finally, the three spatial CV don’t seem so different on one trial. We have resampled the set of folds 15x for you and the results are presented in the following figures (/! epileptics, be careful):


Fold method 1
60
40
20
0
-20
0
20
40


Fold method 2
60
40
20
0
-20
0
20
40



B.2. Model fitting

Spatial cross-validation hasn't been implemented within the `gbm.step` function. However, an function argument (`fold.vector`) can be specified for loading user-supplied fold vector (The `datN1$LabGeoX` vectors). In the following section, one example of Spatial CV is implemented for the quantile method.

```
# Saves the results of BRT
myBRTGeoL <- list()

# Computes 3 BRT, one for each set of predictor variables
for(set in 1:3){

  # Selects the dependent variable. Finds the ID of the column "IsPresent" in dataTo
t.
  myPosDep <- match("IsPresent", names(datN1))

  # Selects the predictors. Finds the ID of predictors in dataTot
  myPosPred <- match(P_VarList[[set]], names(datN1))

  # Attempts to compute the model on the training data
  myBRTGeo <- gbm.step(data = datN1, gbm.x = myPosPred, gbm.y = myPosDep,
                       family = "bernoulli", tree.complexity = P_tc,
                       learning.rate = P_lr,  # /!\ Don't change your learning rate
                       n.folds = P_kfold,
                       n.trees = 5, step.size = 20, # Refines your search for a optim
um number of trees
                       fold.vector = datN1$LabGeo2, # /!\ Sets the folds
                       silent = T, plot.main = F)

  # Saves the total model
  myBRTGeoL[[set]] <- myBRTGeo

}
```

```
## loading user-supplied fold vector
## loading user-supplied fold vector
## loading user-supplied fold vector
```

```
summary(myBRTGeoL)
```

```
##      Length Class Mode
## [1,] 41     gbm   list
## [2,] 41     gbm   list
## [3,] 41     gbm   list
```

```
# Extracts the AUC results from myBRTGeoL and saves it with the previous results
Mauc <- rbind(st = Mauc,
              sp = sapply(1:3, function(ii) myBRTGeoL[[ii]]$cv.statistics["discrimina
tion.mean"][[1]]))

SEauc <- rbind(st = SEauc,
               sp = sapply(1:3, function(ii) myBRTGeoL[[ii]]$cv.statistics["discrimin
ation.se"][[1]]))

Mauc
```

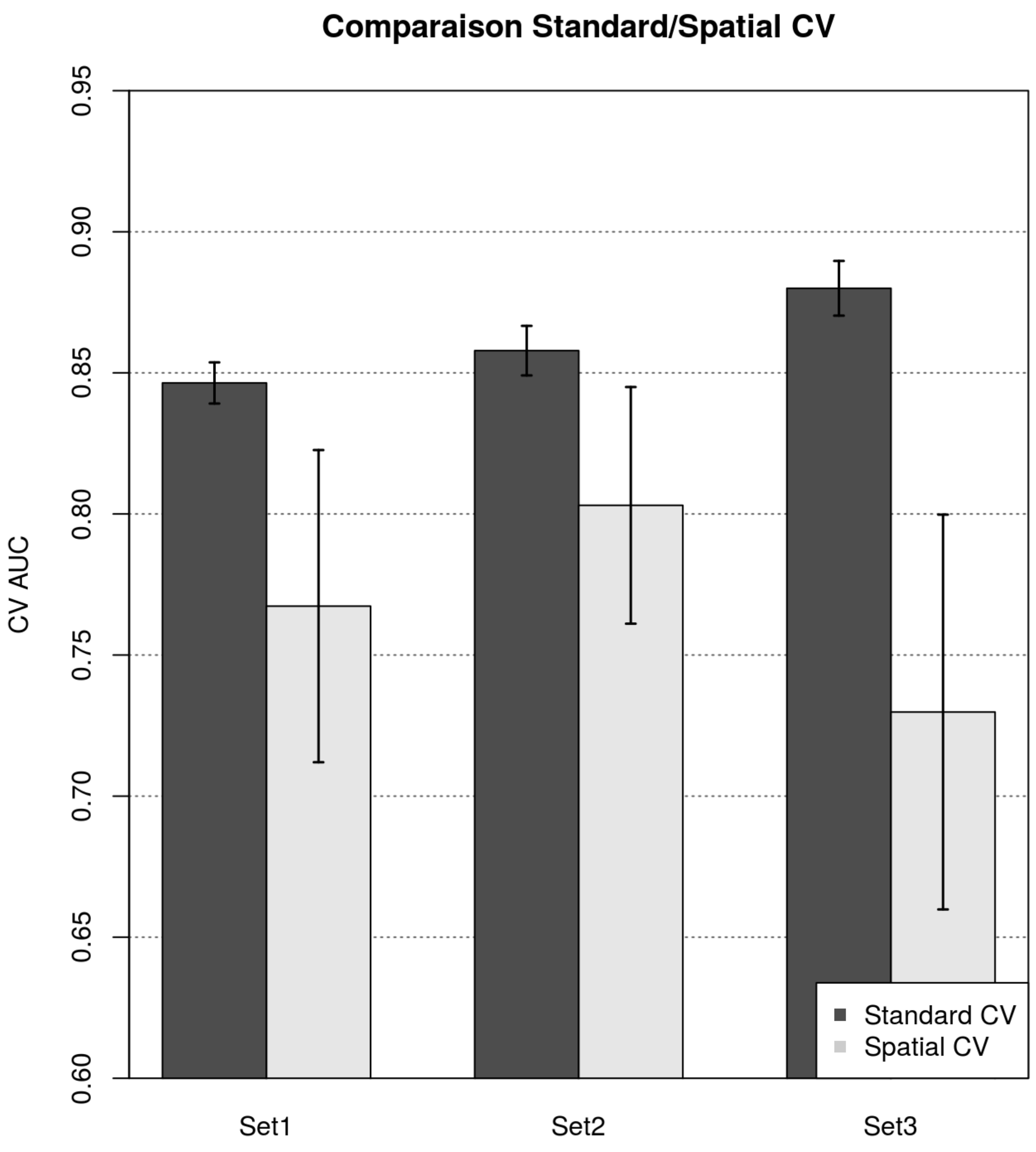
```
##        [,1]     [,2]     [,3]
## st 0.846425 0.857875 0.879975
## sp 0.767325 0.803050 0.729800
```

```
SEauc
```

```
##          [,1]        [,2]        [,3]
## st 0.00729696 0.008778513 0.009698743
## sp 0.05532078 0.041932674 0.069954831
```

For each Set of predictor variables, the mean AUC decreases and the standard deviation increases

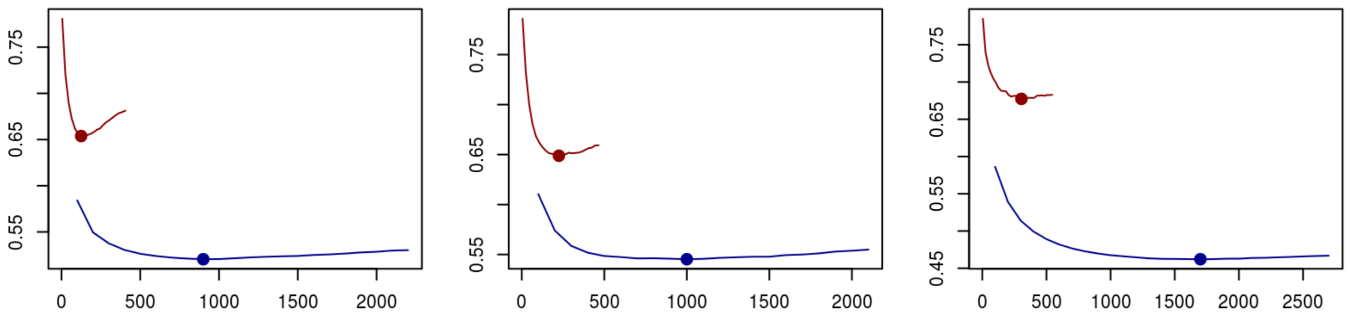
```
# Plots your results
barplot(Mauc, yaxt = "n", ylim = c(0.6, 0.95), border = "white",
        col = "white", beside = TRUE, ylab = "CV AUC",
        names.arg = names(P_VarList),
        main = "Comparaison Standard/Spatial CV")
axis(2, seq(0.5, 1, 0.05))
grid(col = "grey40", nx = 0, ny = 7)
barX <- barplot(Mauc, beside = TRUE, xpd = FALSE, add = T, yaxt = "n")
arrows(barX, Mauc - SEauc, barX,
       Mauc + SEauc, lwd = 1.5, angle = 90,
       code = 3, length = 0.03)
box()
legend("bottomright", legend = c("Standard CV", "Spatial CV"), pch = 15, col = c("gre
y30", "grey80"))
```



For the HPAI H5N1 models, the overall goodness of fit metrics for stantard cv were good with high predictive accuracy AUC values higher than 0.80 (dark grey). From Set 1, the standard AUC increases mainly with the Set 3. However, when goodness of fit was evaluated spatially, the AUC decrease dramatically. Climatic variables (set 3) own the best standard AUC and the worst spatial AUC showing the highest ability to overfit.

B.3.b. Holdout deviance

```
HdevGG <- list()

par(mfrow = c(1, 3), mar = rep(2, 4))
for(set in 1:3){

  # Extracts the BRT results
  cvSt <- SauvBrt[[set]]$cv.values
  cvSp <- myBRTGeoL[[set]]$cv.values

  ntSt <- SauvBrt[[set]]$trees.fitted
  ntSp <- myBRTGeoL[[set]]$trees.fitted

  limX <- range(c(ntSt, ntSp))
  limY <- range(c(cvSt, cvSp))

  # Plots the holdout deviance as a function of the number of trees
  plot(y = cvSt, x = ntSt, type = "l", col = "darkblue",
       xlim = limX, ylim = limY, xlab = "Number of trees", ylab = "Holdout deviance"
)
  points(y = cvSp, x = ntSp, type = "l", col = "darkred")

  # Adds the optimal number of trees
  OptSt <- SauvBrt[[set]]$gbm.call$best.trees
  OptSp <- myBRTGeoL[[set]]$gbm.call$best.trees

  points(y = cvSt[which(ntSt == OptSt)], x = OptSt, col = "darkblue", cex = 2, pch =
20)
  points(y = cvSp[which(ntSp == OptSp)], x = OptSp, col = "darkred", cex = 2, pch = 2
0)

}
```



C. Accounting for Spatial Sorting Bias

C.1. Definition

The following section is taken from Hijmans (2012) (http://www.jstor.org/stable/23143954?seq=1#page_scan_tab_contents):

Because of spatial autocorrelation, the environment (climate in particular) of nearby sites is expected to more similar than that of distant sites. It follows that the nearer testing-presence sites are to training-presence sites the more similar their environments, and the higher the predicted suitability of those sites will be. Likewise, the more distant the testing-absence sites from the training-presence sites, the lower the model predictions for those sites will be.

The distance between training-presence and testing-presence sites will tend to be smaller than the distance *between training-presence and testing-absence sites. I refer to this phenomenon of* ‘attraction’ *of testing-presence sites and* ‘repulsion’ *of testing-absence sites to presence-training sites as* ‘spatial sorting bias’ *(SSB). This bias is likely very common, such that cross-validation results should generally be inflated.*

Hijmans suggested one metric to measure the SSB (the ‘3.C.2. Measuring the SSB intensity’ section) and two corrected AUC: *I propose the use of pairwise distance sampling to remove this bias* (the ‘3.C.3. Sorting testing-data’ section)*, and the use of a null model that only considers the geographic distance to training sites to calibrate cross-validation results for remaining biais*.

C.2. Measuring the intensity of SSB

Hijmans suggested a metric to measure the intensity of SSB:

SSB = Dp / Da

Dp = the mean distance of testing-presence sites to the nearest training-presence site
Da = the mean distance of testing-absence sites to the nearest training-presence site

SSB = 1 suggests there is no spatial sorting bias
SSB = 0 indicates extreme spatial sorting biais

```
### First, computes the SSB over the standard cv dataset

# Splits the dataset in training/testing-data by means
# of two fold vector (dataTot$LabAlea and dataTot$LabGeo)
idTr1 <- which(datN1$IsPresent == 1 & datN1$LabAlea != 3)
idTr0 <- which(datN1$IsPresent == 0 & datN1$LabAlea != 3)

idVal1 <- which(datN1$IsPresent == 1 & datN1$LabAlea == 3)
idVal0 <- which(datN1$IsPresent == 0 & datN1$LabAlea == 3)

Tr1 <- datN1[idTr1,]
Tr0 <- datN1[idTr0,]

Val1 <- datN1[idVal1,]
Val0 <- datN1[idVal0,]

# Computes the SSB with the 'ssb' from 'dismo' package
SSB1 <- ssb(Val1[, c("x","y")], Val0[, c("x","y")],
            Tr1[, c("x","y")], lonlat = TRUE)

### Second, computes the SSB over the geographic cv dataset
idTr1geo <- which(datN1$IsPresent == 1 & datN1$LabGeo2 != 3)
idTr0geo <- which(datN1$IsPresent == 0 & datN1$LabGeo2 != 3)

idVal1geo <- which(datN1$IsPresent == 1 & datN1$LabGeo2 == 3)
idVal0geo <- which(datN1$IsPresent == 0 & datN1$LabGeo2 == 3)

Tr1geo <- datN1[idTr1geo,]
Tr0geo <- datN1[idTr0geo,]

Val1geo <- datN1[idVal1geo,]
Val0geo <- datN1[idVal0geo,]

SSB2 <- ssb(Val1geo[, c("x","y")], Val0geo[, c("x","y")],
            Tr1geo[, c("x","y")], lonlat = TRUE)

# Compares the results
# Dp and Da
SSB1 # Standard CV
```

```
##             p        a
## [1,] 38023.76 699328.8
```

```
SSB2 # Spatial CV
```

```
##             p       a
## [1,] 530728.1 1479076
```

```
# The SSB metric
SSB1[1, "p"] / SSB1[1, "a"]
```

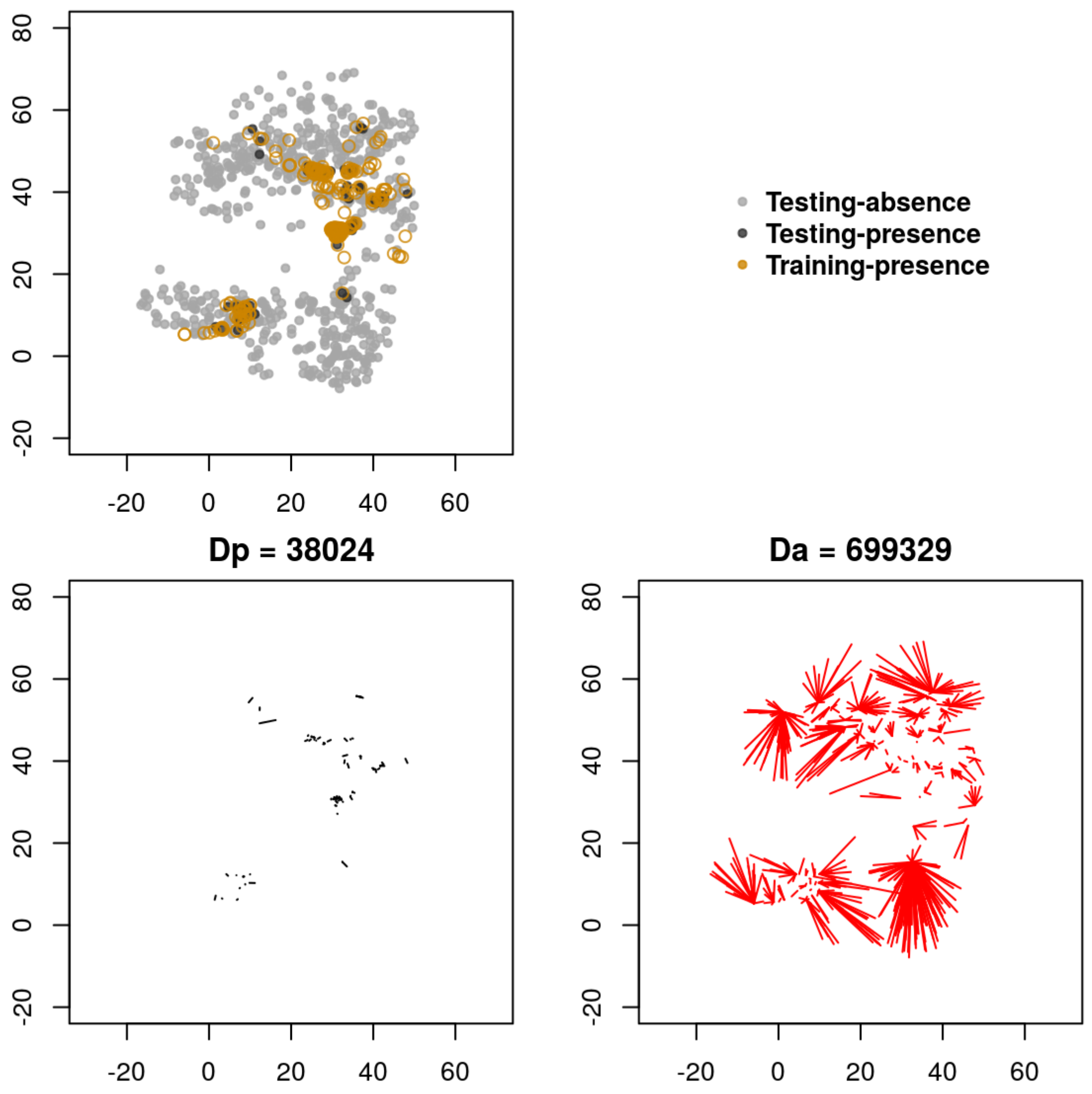
```
##          p
## 0.05437178
```

```
SSB2[1, "p"] / SSB2[1, "a"]
```

```
##         p
## 0.3588241
```

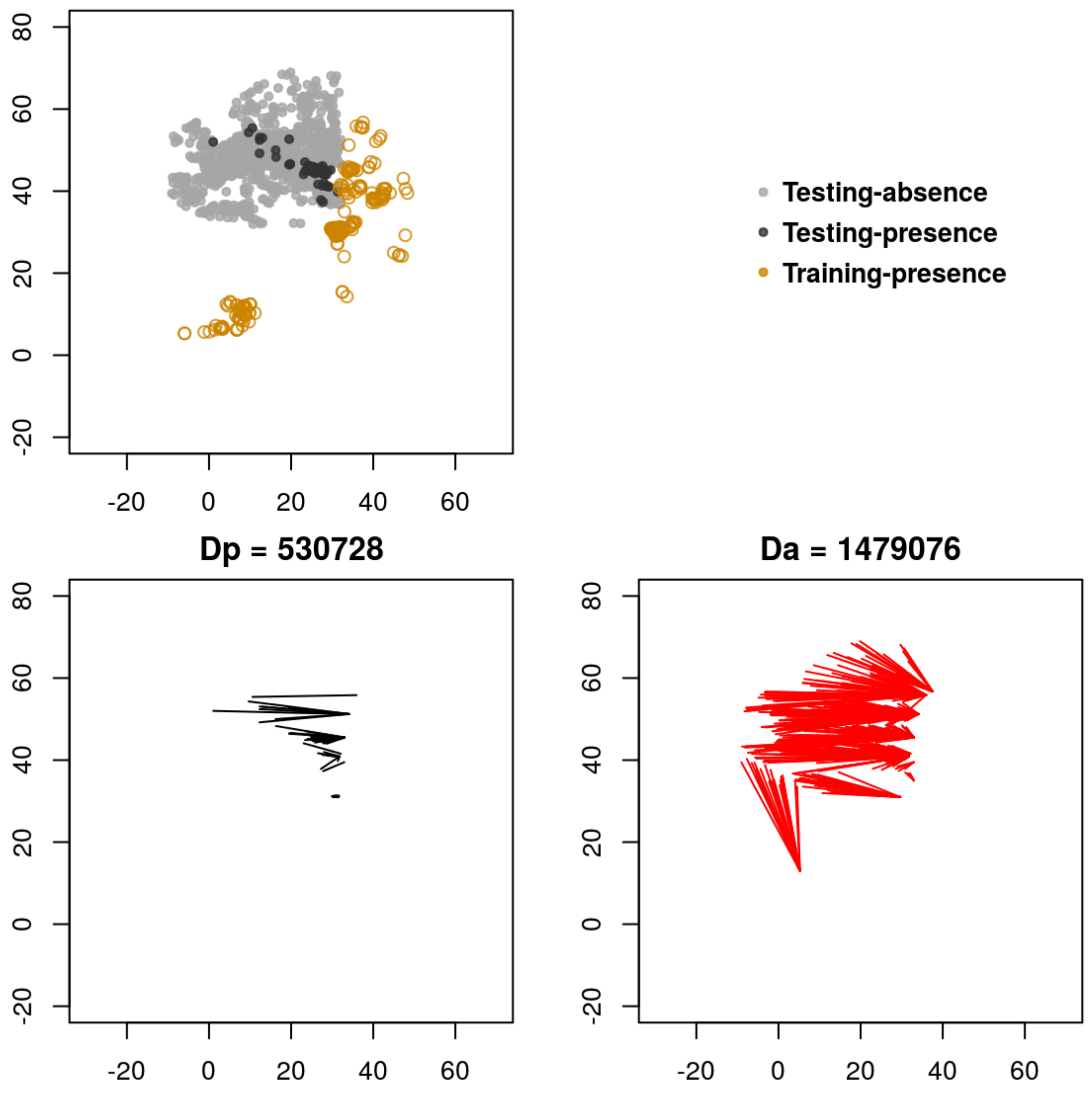
The SSB was high for the standard CV (close to 0) and the SSB decrease greatly with the establishment of a geographic cross-validation. Now, we will try to illustrate the SSB with some maps. On following maps, the distance of testing-presence or testing-absence sites to the nearest training-presence site is explicit.

```r
# Identifies the nearest training-presences of testing-data
# (presences and absences separately)
idDp <- Lib_DistEstimatesID(as.matrix(Val1[, c("x","y")]), as.matrix(Tr1[, c("x","y")
]))
idDa <- Lib_DistEstimatesID(as.matrix(Val0[, c("x","y")]), as.matrix(Tr1[, c("x","y")
]))

### Builds some maps for representation

# Specify a new colour palette
P_Col_fig2 <- c(rgb(0.65, 0.65, 0.65, 0.8),
                rgb(0.2, 0.2, 0.2, 0.8),
                rgb(205, 133, 0, 204, max = 255))



# Splits the plot-window in four: two rows and two columns
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))

# Plots the testing-absence dataset in light grey
plot(Val0[,c("x","y")], col = P_Col_fig2[1],
     pch = 20, xlim = c(-30, 70), ylim = c(-20, 80), asp = 1)
# Plots the testing-presence dataset in dark grey
points(Val1[,c("x","y")], col = P_Col_fig2[2], pch = 20)
# Plots the training-presence dataset in orange
points(Tr1[,c("x","y")], col = P_Col_fig2[3])

# Plots the caption
plot(1:10, 1:10, type="n", axes=F, xlab="", ylab="")
legend("center", pch = 20, c("Testing-absence", "Testing-presence", "Training-presenc
e"),
       col = P_Col_fig2, text.font = 2, cex = 1, bty = "n")

# Writes a title for the plot
titl <- paste0("Dp = ", round(SSB1[1, "p"]))

# First, plots the point included in the testing-presence
# and the corresponding nearest training-presence.
# Thereafter, draws a line that connects the two points.
plot(rbind(Val1[1,c("x","y")], Tr1[idDp[1],c("x","y")]),
     ylim = c(-20, 80),
     xlim = c(-30, 70),
     type ="l", main = titl, asp = 1)

# Repeats the process for each row of the testing-presence
for(ii in 2:nrow(Val1)){
  points(rbind(Val1[ii,c("x","y")],
               Tr1[idDp[ii],c("x","y")]),
         type ="l")
}

# Writes a title for the plot
titl <- paste0("Da = ", round(SSB1[1, "a"]))

# Repeats the process for the testing-absence
plot(rbind(Val0[1, c("x","y")], Tr1[idDp[1], c("x","y")]),
     ylim = c(-20, 80),
     xlim = c(-30, 70),
     type ="l", main = titl, col = "red", asp = 1)

for(ii in 2:nrow(Val0)){
  points(rbind(Val0[ii, c("x","y")],
               Tr1[idDa[ii], c("x","y")]),
         col="red", type ="l")
}
```



At the bottom left, the black lines represent the distance of testing-presence sites to the nearest training-presence site. At the bottom left, the red lines represent the distance of testing-absence sites to the nearest training-presence site. Overall, the red lines appear longer than the black lines. Therefore, the mean distance Da is higher than the mean distance Dp and the SSB metric approaches 0.

```
# The same process and the same plot is repeated for the geographic training/testing-
data
MyP
```



After the spatial cross-validation, the sizes of the red and black lines appear more balanced.

C.3. Sorting testing-data

These models are also evaluated after sub-sampling the testing data using pairwise distance sampling to attempt to remove spatial sorting bias from the testing data, using the pwdSample function in dismo package. The first step in this approach is to compute, for each testing-presence site, the distance to the nearest training-presence site. Each testing-presence site is paired with the testing-absence site that has the most similar distance to its nearest training-presence site. If the difference between the two distances is more than a specified threshold (I used 33%) the presence site is not used. Hijmans (2012).

```
# This example was developed over the Set 1
myPosPred <- match(P_VarList[[1]], names(datN1))
myPosDep <- match("IsPresent", names(datN1))

# Runs the BRT on the training data
myBRTssb <- gbm.step(data = rbind(Tr1,Tr0), gbm.x = myPosPred, gbm.y = myPosDep,
                     family = "bernoulli", tree.complexity = 4, learning.rate = 0.01,
                     bag.fraction = 0.5, n.folds = 4, n.trees = 200, step.size = 100,
                     silent = T, plot.main = F)

# Sub-samples the test-presences by means of the Hijmans procedure
i <- pwdSample(Val1[,c("x","y")], Val0[,c("x","y")], Tr1[,c("x","y")],
               lonlat = TRUE, nearest = TRUE)

sVal1 <- Val1[!is.na(i), ]

# Generates the new testing-data, filtered to reduce the SSB
ValDF <-  rbind(sVal1,Val0)

# Checks the initial number of presence/absence in the testing-data
ValDFb <-  rbind(Val1, Val0)
table(ValDFb$IsPresent)
```

```
##
##   0   1
## 600  87
```

```
# Checks the number of presence/absence in the filtered testing-data
table(ValDF$IsPresent)
```

```
##
##   0   1
## 600  37
```

```
# Predicts the presence/absence of virus over the filtered testing-data
ValDF$myPred1 <- predict.gbm(myBRTssb, ValDF, type="response",
                             n.trees=myBRTGeo1$gbm.call$best.trees)

# Now, compares the two AUC
# The standard AUC
myBRTssb$cv.statistics$discrimination.mean
```

```
## [1] 0.824025
```

```
# The filtered AUC of Hijmans computed with the 'roc' function from the pROC package
roc(ValDF$IsPresent ~ ValDF$myPred1, data = ValDF)$auc[1]
```

```
## [1] 0.7685586
```

The filtered AUC of Hijmans decreases in agreement with our previous result (the presence of SSB over the standard CV dataset).

IV. References

1. Dhingra, Madhur S., et al. "Global mapping of highly pathogenic avian influenza H5N1 and H5Nx clade 2.3. 4.4 viruses with spatial cross-validation." eLife 5 (2016): e19571.
2. Wenger SJ, Olden JD. Assessing transferability of ecological models: an underappreciated aspect of statistical validation. Methods in Ecology and Evolution. 2012 Apr 1;3(2):260-7.
3. Elith J, Leathwick JR, Hastie T. A working guide to boosted regression trees. Journal of Animal Ecology. juill 2008;77(4):802-13.
4. Jiménez-Valverde A. Insights into the area under the receiver operating characteristic curve (AUC) as a discrimination measure in species distribution modelling. Global Ecology and Biogeography. 2012 Apr 1;21(4):498-507.
5. Muscarella, Robert, et al. "ENMeval: an R package for conducting spatially independent evaluations and estimating optimal model complexity for Maxent ecological niche models." Methods in Ecology and Evolution 5.11 (2014): 1198-1205.
6. Hijmans RJ. Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null model. Ecology. 2012 Mar;93(3):679-88.

V. Functions

A. 'thrSample'

```r
##########################
## 'thrSample' function ##
##########################
thrSample <- function(
  thr = 4000, # the minimum distance
  coords = c("x", "y"), # the column names of geographic coordinates
  data = data, # the dataset
  nPts = 4) { # the number of items to choose

  # Stocks the results in 'MyId'
  MyId <- rep(NA, nPts)

  # Samples one observation from 'data' and saves it
  MyRow <- list(1:nrow(data))
  MyId[1] <- sample(MyRow[[1]], 1)

  # Computes all the distances between the sampled
  # observation and the remaining observations
  # and stores it in 'MyDist'
  MyDist <- list(spDistsN1(as.matrix(data[,coords]),
                           as.matrix(data[MyId[1], coords]), longlat = TRUE))

  # Repeats the previous process a number of time
  # equal to the number of items to choose
  # -1 (to account for the first sampled observation)
  for(ii in 2:nPts){

    # Selects the ID of row that meet the minimum distance condition
    MyRowTr <- which(MyDist[[(ii-1)]] > thr)

    # Stops the function if no observation meets the condition
    if(length(MyRowTr) == 0) {
      stop("Restarts the function with a smaller minimum distance")
    }

    # Stores the ID
    MyRow[[ii]] <- MyRowTr

    # Counts the number of time the observations meet the
    # distance condition over iteration loops already performed
    TestTabl <- table(unlist(MyRow))

    # Samples an observation from idThr, a vector of observation
    # that meet each time the minimum distance condition
    idThr <- as.numeric(names(which(TestTabl == ii)))
    MyId[ii] <- sample(idThr, 1)

    # Computes the new distances
    MyDist[[ii]] <- spDistsN1(as.matrix(data[, coords]),
                              as.matrix(data[MyId[ii], coords]), longlat = TRUE)
  }
  return(MyId)
}
```

B. 'Lib_DistEstimatesID'

```
####################################
## 'Lib_DistEstimatesID' function ##
####################################
# Returns the identity row of nearest point
# in a vector Y for each element of a vector X.

Lib_DistEstimatesID <- function(myVec1, myVec2, longlat = TRUE){
  MyRes <- sapply(1:nrow(myVec1), function(i)
    which.min(spDistsN1(myVec2, myVec1[i,], longlat = longlat)))
  return(MyRes)
}
```

C. ‘get.block’

From ‘ENMeval’ package

```r
## https://github.com/cran/ENMeval/tree/master/R
get.block <- function(occ, bg.coords){
  # SPLIT OCC POINTS INTO FOUR SPATIAL GROUPS
  noccs <- nrow(occ)
  n1 <- ceiling(nrow(occ)/2)
  n2 <- floor(nrow(occ)/2)
  n3 <- ceiling(n1/2)
  n4 <- ceiling(n2/2)
  grpA <- occ[order(occ[, 2]),][1:n1,]
  grpB <- occ[rev(order(occ[, 2])),][1:n2,]
  grp1 <- grpA[order(grpA[, 1]),][1:(n3),]
  grp2 <- grpA[!rownames(grpA)%in%rownames(grp1),]
  grp3 <- grpB[order(grpB[, 1]),][1:(n4),]
  grp4 <- grpB[!rownames(grpB)%in%rownames(grp3),]

  # SPLIT BACKGROUND POINTS BASED ON SPATIAL GROUPS
  bvert <- mean(max(grp1[, 1]), min(grp2[, 1])) + 0.5
  tvert <- mean(max(grp3[, 1]), min(grp4[, 1])) + 0.5
  horz <- mean(max(grpA[, 2]), min(grpB[, 2])) + 0.5
  bggrp1 <- bg.coords[bg.coords[, 2] <= horz & bg.coords[, 1]<bvert,]
  bggrp2 <- bg.coords[bg.coords[, 2] < horz & bg.coords[, 1]>=bvert,]
  bggrp3 <- bg.coords[bg.coords[, 2] > horz & bg.coords[, 1]<=tvert,]
  bggrp4 <- bg.coords[bg.coords[, 2] >= horz & bg.coords[, 1]>tvert,]

  rr <- bg.coords[bg.coords[, 2] > horz & bg.coords[, 1] <= bvert & bg.coords[, 1] >=
tvert,]

  r <- data.frame()
  if (nrow(grp1) > 0) grp1$grp <- 1; r <- rbind(r, grp1)
  if (nrow(grp2) > 0) grp2$grp <- 2; r <- rbind(r, grp2)
  if (nrow(grp3) > 0) grp3$grp <- 3; r <- rbind(r, grp3)
  if (nrow(grp4) > 0) grp4$grp <- 4; r <- rbind(r, grp4)
  occ.grp <- r[order(as.numeric(rownames(r))),]$grp

  bgr <- data.frame()
  if (nrow(bggrp1) > 0) bggrp1$grp <- 1; bgr <- rbind(bgr, bggrp1)
  if (nrow(bggrp2) > 0) bggrp2$grp <- 2; bgr <- rbind(bgr, bggrp2)
  if (nrow(bggrp3) > 0) bggrp3$grp <- 3; bgr <- rbind(bgr, bggrp3)
  if (nrow(bggrp4) > 0) bggrp4$grp <- 4; bgr <- rbind(bgr, bggrp4)
  bg.grp <- bgr[order(as.numeric(rownames(bgr))),]$grp

  out <- list(occ.grp=occ.grp, bg.grp=bg.grp)
  return(out)
}
```