Species Distribution Modelling with R

Babak Naimi

27 November 2018

sdm package in R

In this exercice, you will use the sdm package in R to fit species distribution models, evaluate their accuracy, and use them to predict probability of occurrence for species in new geographical area (space) or a new time (e.g., future time).

The sdm package put a chain of processes/methods into a workflow that facilitates running different steps in species distribution modelling. These steps include data preparation (data cleaning), model fitting, model evaluation, prediction, ensemble forecasting, and some post-processing procedures.

While you go through this document, you can review the concepts by looking at the slides presented in the class whenever it is necessary that would be helpful for a better understanding!

Before starting this exercise, you need to have R and RStudio installed on your machine, then from RStudio, install the sdm package, and finally use the following code to install all the other required packages (in case if you don't have them installed on your machine).

You can install the package from CRAN using the function: install.packages("sdm")

BUT, sometimes the CRAN version is not the latest one because the package is actively under development.

You can make sure to have the latest version of the package installed if you use the version on the GitHub repository. Following steps can be taken to install the sdm package from GitHub:

To install any package from GitHub, you can use install_github function in the devtools package. So, you first need to install the devtools package:

```
install.packages('devtools')
```

Then, you can install the sdm from GitHub using:

```
library(devtools)
```

install_github('babaknaimi/sdm')

load the library

```
library(sdm)
```

The sdm package supports several modelling methods most of which depend on the other packages. Therefore, you need to, first, install all the required packages for the methods supported by sdm. I make it easy to install these packages by adding a simple function (installAll) in the sdm package to do the task:

```
installAll()
```

In the Data folder, you can find the datasets that are used in this exercise.

```
library(raster)
```

```
## Loading required package: sp
```

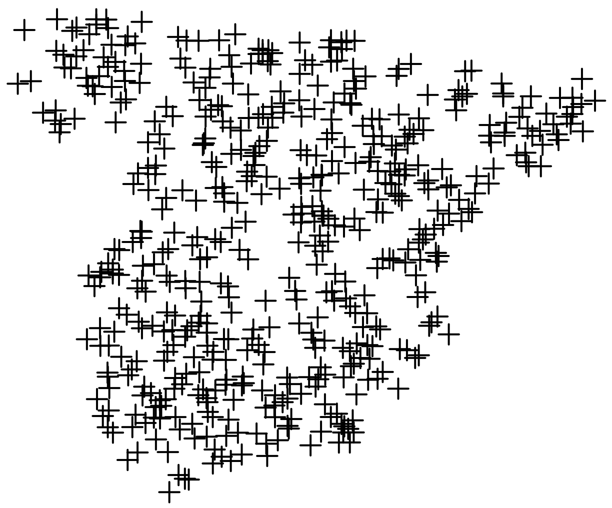
```
library(rgdal)
```

```
## rgdal: version: 1.3-3, (SVN revision 759)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.1.3, released 2017/20/01
##  Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.5/Resources/library/rgdal/gdal
##  GDAL binary built with GEOS: FALSE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.5/Resources/library/rgdal/pr
##  Linking to sp version: 1.3-1
```

```
sp <- shapefile('PA_Mauremys leprosa.shp') # loading shapefile (SpatialPointsDataFrame)
```

```
## Warning in .local(x, ...): .prj file is missing
```

```
plot(sp)
```



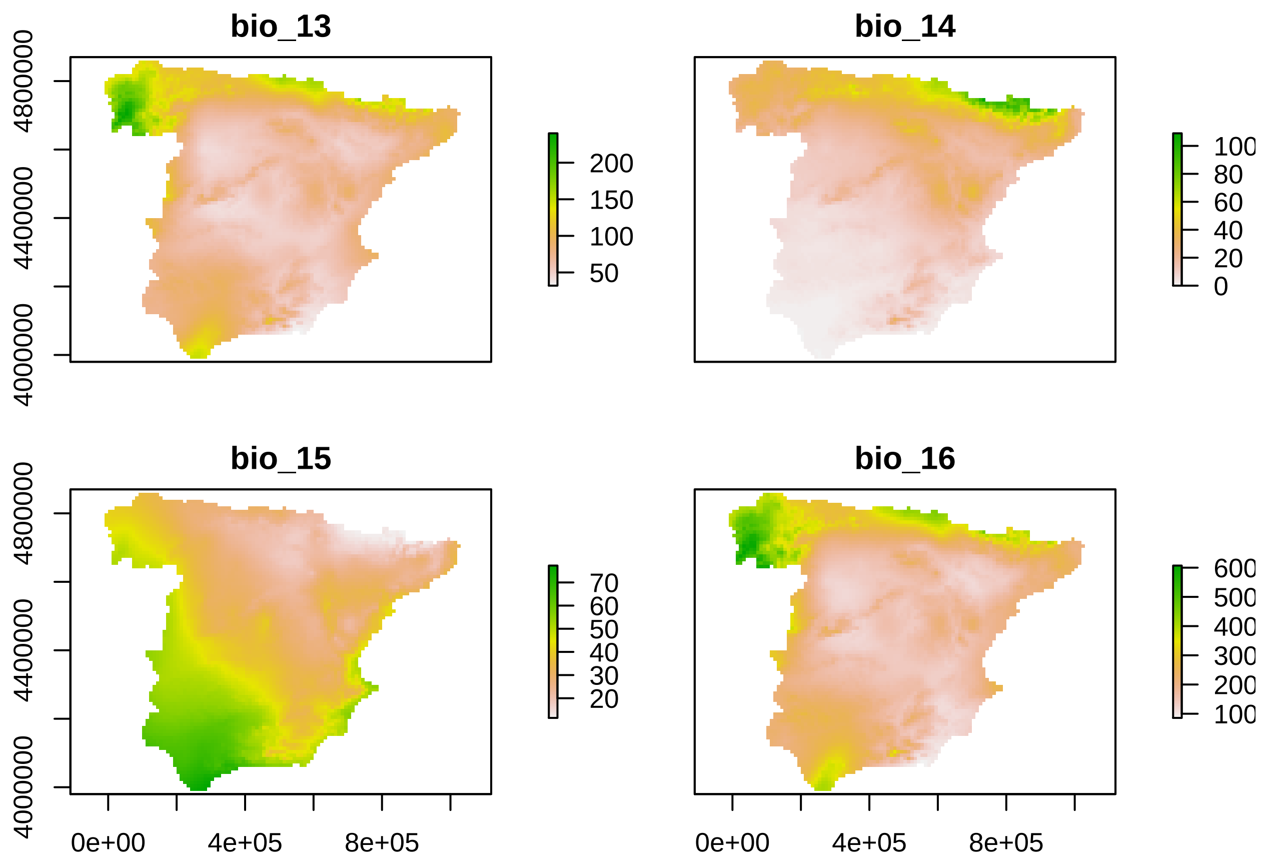
```
lst <- list.files(pattern='asc$') # list the name of files in working directory that ends to asc

lst # this is the name of raster files we want to use as predictor variables
```

```
##  [1] "bio_1.asc"  "bio_10.asc" "bio_11.asc" "bio_12.asc" "bio_13.asc"
##  [6] "bio_14.asc" "bio_15.asc" "bio_16.asc" "bio_17.asc" "bio_18.asc"
## [11] "bio_19.asc" "bio_2.asc"  "bio_3.asc"  "bio_4.asc"  "bio_5.asc"
## [16] "bio_6.asc"  "bio_7.asc"  "bio_8.asc"  "bio_9.asc"
```

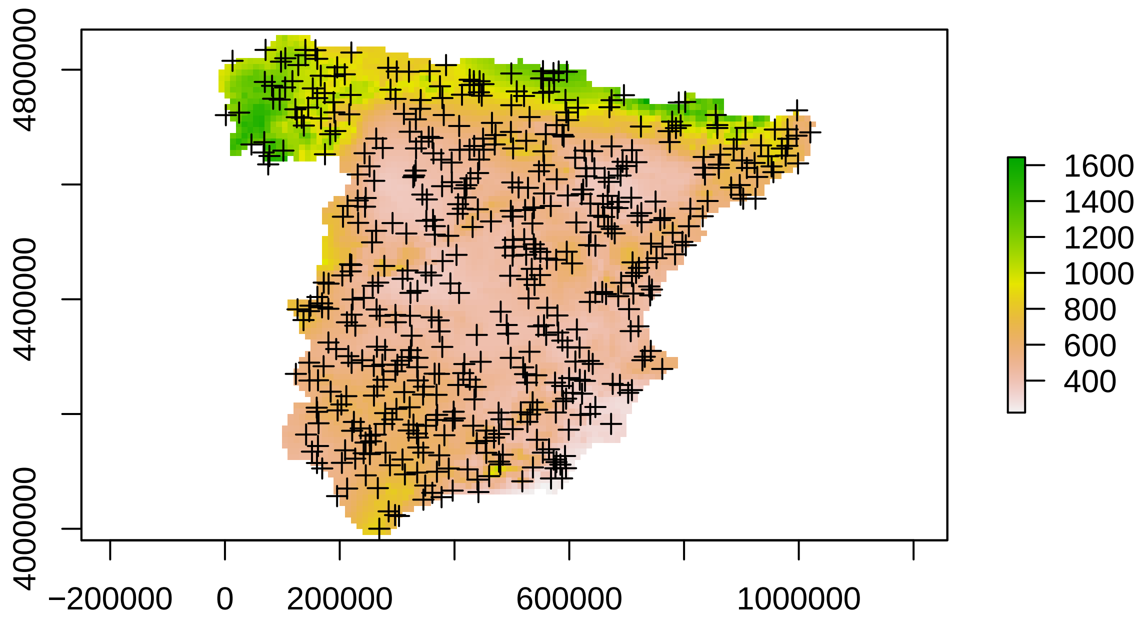
```
preds <- stack(lst) # making a raster object

preds # see the specification of the raster layers (e.g., cell size, extent, etc.)
```

```
## class       : RasterStack
## dimensions  : 89, 105, 9345, 19  (nrow, ncol, ncell, nlayers)
## resolution  : 10000, 10000  (x, y)
## extent      : -20625, 1029375, 3979936, 4869936  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=30 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
## names       : bio_1, bio_10, bio_11, bio_12, bio_13, bio_14, bio_15, bio_16, bio_17, bio_18, bio_19,
```

```r
plot(preds[[5:8]]) # only plots layes 5 to 8
```



```r
plot(preds[[4]]) # only plot the 4th layer

plot(sp,add=T) # let's add the species point on the previous plot
```



Data preparation

We first need to check if the data are subjected to the multicollinearity issue. You can use the function vifstep (or vifcor) that I developed in the usdm package to find out which variables should be excluded due to the collinearity issue. vifstep reports the status of the variables (their VIF statistic) followed by a list of variables that should be excluded. We can then use the exclude function, to physically exclude those variables from the main dataset.

```r
library(usdm) # a specific library I developed to address the problems with SDMs
# (i.e., uncertainty, collinearity)

# As the input of the function, you can use either the Raster* dataset (predictors)
# or a data.frame contains the predictor variables
v <- vifstep(preds)

v # see the report
```

```
## 12 variables from the 19 input variables have collinearity problem:
##
## bio_5 bio_10 bio_1 bio_7 bio_12 bio_16 bio_11 bio_17 bio_18 bio_2 bio_19 bio_14
##
## After excluding the collinear variables, the linear correlation coefficients ranges between:
## min correlation ( bio_9 ~ bio_8 ):  -0.02787882
## max correlation ( bio_4 ~ bio_13 ):  -0.7346922
##
## ---------- VIFs of the remained variables --------
##   Variables      VIF
## 1    bio_13 3.687064
## 2    bio_15 4.307036
## 3     bio_3 1.614461
## 4     bio_4 6.066428
## 5     bio_6 7.462489
## 6     bio_8 2.440162
## 7     bio_9 2.652470
```

```r
# now, you can exclude those that have the collinearity issue:

preds <- exclude(preds) # we overwrite on the original predictors dataset (i.e., preds)
```

```
## 12 variables from the 19 input variables have collinearity problem:
##
## bio_5 bio_10 bio_11 bio_7 bio_12 bio_16 bio_6 bio_17 bio_18 bio_4 bio_13 bio_14
##
## After excluding the collinear variables, the linear correlation coefficients ranges between:
## min correlation ( bio_9 ~ bio_8 ):  -0.02364514
## max correlation ( bio_15 ~ bio_1 ):  0.7722403
##
## ---------- VIFs of the remained variables --------
##   Variables      VIF
## 1     bio_1 7.566743
## 2    bio_15 4.802661
## 3    bio_19 4.103620
## 4     bio_2 2.945471
## 5     bio_3 1.306032
## 6     bio_8 3.652808
## 7     bio_9 2.753151
```

```
#---
preds
```

```
## class       : RasterStack
## dimensions  : 89, 105, 9345, 7  (nrow, ncol, ncell, nlayers)
## resolution  : 10000, 10000  (x, y)
## extent      : -20625, 1029375, 3979936, 4869936  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=30 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
## names       : bio_1, bio_15, bio_19, bio_2, bio_3, bio_8, bio_9
```

Data preparation for SDM

Now, both the species and predictor data are available. The first step to use the sdm package is to create a data object using the **sdmData** function. In this function, you can introduce a train dataset (can be either spatial points or simply a data.frame), predictor variables (if available separately as a Raster* object), a test dataset (ONLY if an independent test dataset is available for validation). In addition, you can specify the name(s) of species and predictor variables in a formula. If the formula is not specified, the function tries to detect which variables contain species data and consider the rest as predictor variables.

```
library(sdm)
# we need to know the name of species column in the species dataset (i.e., sp object)

head(sp) # it shows the first few rows of the attribute table linked to the species points

d <- sdmData(Occurrence~., train=sp, predictors = preds)
```

```
d
```

```
## class                                 : sdmdata
## ===========================================================
## number of species                     :  1
## species names                         :  Occurrence
## number of features                    :  7
## feature names                         :  bio_1, bio_15, bio_19, ...
## type                                  :  Presence-Absence
## has independet test data?             :  FALSE
## number of records                     :  512
## has Coordinates?                      :  TRUE
```

In the above example, we used a species dataset that contains both presence and absence records. In many situations, species data have only presence records. To be able to use most of modelling methods, we need both presence and absence. A solution to deal with presence-only data is to generate some randomly drawn points and use them as absence records (called pseudo-absence or background). In the sdmData function, the bg argument can be added to generate background data.

d <- sdmData(Occurrence~., train=sp, predictors = preds, bg=list(n=1000)) # size of background is 1000.

Model Fitting

When the sdmdata object is created (d in the above example), you would be able to fit the models. To do so, you can use the **sdm** function. In this function, you define a formula (specifies the structure of the model), data, methods, and settings for different steps of the sdm workflow (e.g., evaluation, models, etc.).

```r
getmethodNames() # shows the list of methods supports by the sdm package installed on your machine
```

```
## $bioclim
## [1] "bioclim" "Bioclim"
##
## $bioclim.dismo
## [1] "bioclim.dismo" "BioclimDismo"  "bioclimDismo"  "bioclim.dis"
## [5] "bioclimD"
##
## $brt
## [1] "brt" "BRT" "gbm" "GBM"
##
## $cart
## [1] "cart" "CART" "tree"
##
## $domain.dismo
## [1] "domain.dismo" "DomainDismo"  "domainDismo"  "domain.dis"
## [5] "domainD"
##
## $fda
## [1] "fda" "FDA"
##
## $gam
## [1] "gam" "GAM"
##
## $glm
## [1] "glm" "GLM" "lm"
##
## $glmnet
## [1] "glmnet"     "GLMNET"     "glmelastic" "glmlasso"
##
## $mahal.dismo
## [1] "mahal.dismo"       "MahalDismo"        "mahalanoubisDismo"
## [4] "mahal.dis"         "mahalD"
##
## $mars
## [1] "mars"  "MARS"  "earth"
##
## $maxent
## [1] "maxent"     "Maxent"     "MAXENT"     "entropy"    "maxentropy"
##
## $maxlike
## [1] "maxlike" "MaxLike"
##
## $mda
## [1] "mda" "MDA"
##
## $mlp
## [1] "mlp"      "MLP"      "nnet.mlp" "nnetMLP"
##
## $rbf
## [1] "rbf"      "RBF"      "nnet.rbf" "nnetRbf"
##
## $rf
## [1] "rf"           "RF"           "randomForest" "rforest"
##
## $rpart
## [1] "rpart" "RPART"
##
## $svm
## [1] "svm"  "SVM"  "ksvm"
```

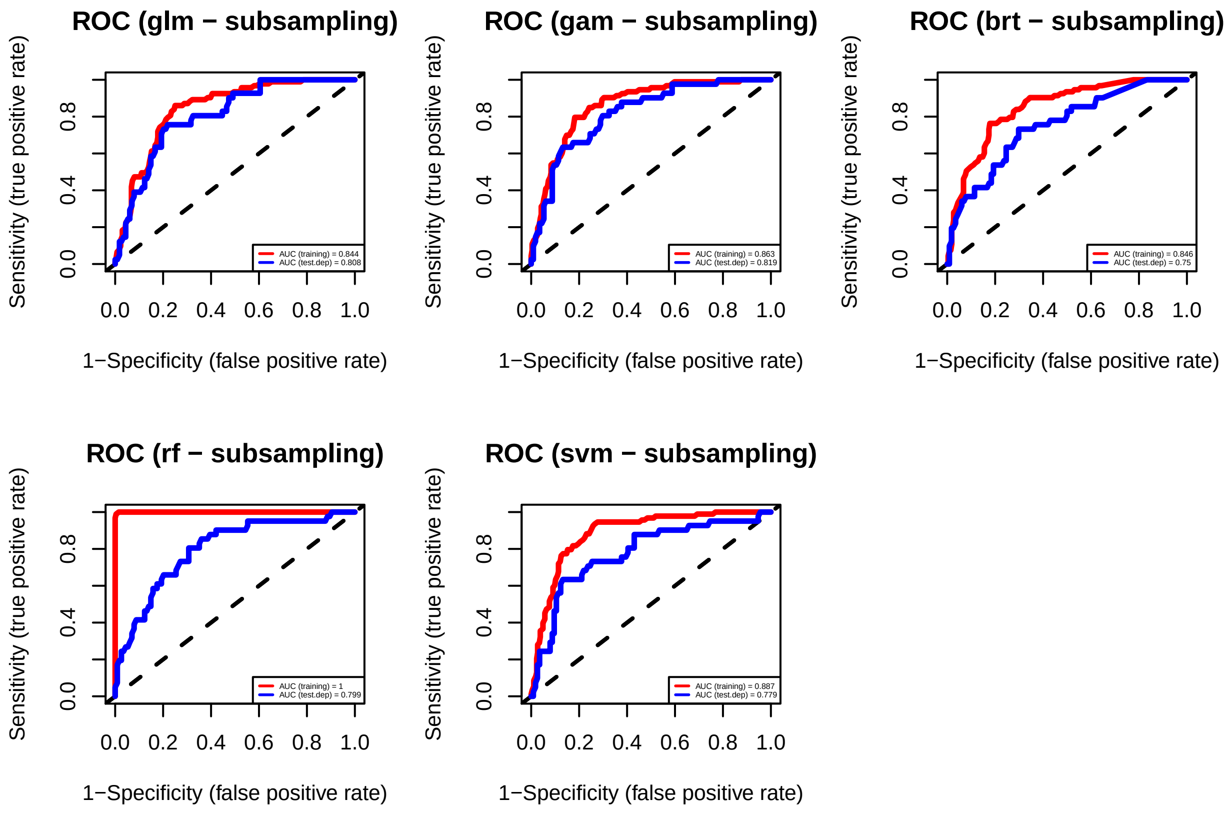
```
# in the following example, we use 5 different methods to fit the models. We also asked the function to

m1 <- sdm(Occurrence~.,data=d,methods=c('glm','gam','brt','rf','svm'),test.percent=30)
```

```
##
## Attaching package: 'mmap'

## The following object is masked from 'package:Rcpp':
##
##     sizeof
```

```
#-------------------

# this is the object containing the models, and here when you see it,
#it generates a brief report:
m1
```

```
## class                                 : sdmModels
## ========================================================
## number of species                     :  1
## number of modelling methods           :  5
## names of modelling methods            :  glm, gam, brt, rf, svm
## replicate.methods (data partitioning) :  subsampling
## number of replicates (each method)    :  1
## toral number of replicates per model  :  1 (per species)
## test percentage (in subsampling)      :  30
## ------------------------------------------
## model run success percentage (per species)  :
## ------------------------------------------
## method          Occurrence
## ----------------------
## glm        :        100   %
## gam        :        100   %
## brt        :        100   %
## rf         :        100   %
## svm        :        100   %
##
## ###################################################################
## model performance (per species), using test dataset (generated using partitioning):
## --------------------------------------------------------------------------------
##
##  ## species   :  Occurrence
## =========================
##
## methods    :     AUC     |     COR     |     TSS     |     Deviance
## -------------------------------------------------------------------------
## glm        :     0.81    |     0.48    |     0.54    |     0.92
## gam        :     0.82    |     0.5     |     0.52    |     0.9
## brt        :     0.75    |     0.38    |     0.43    |     1.02
## rf         :     0.8     |     0.48    |     0.5     |     0.95
## svm        :     0.78    |     0.47    |     0.5     |     1.29
```

```
# the report contains the information on whether the models are successfully run,
# as well as the accuracy based on some statistics

# you can generate the roc curve and compare the results for all models:
roc(m1)
```

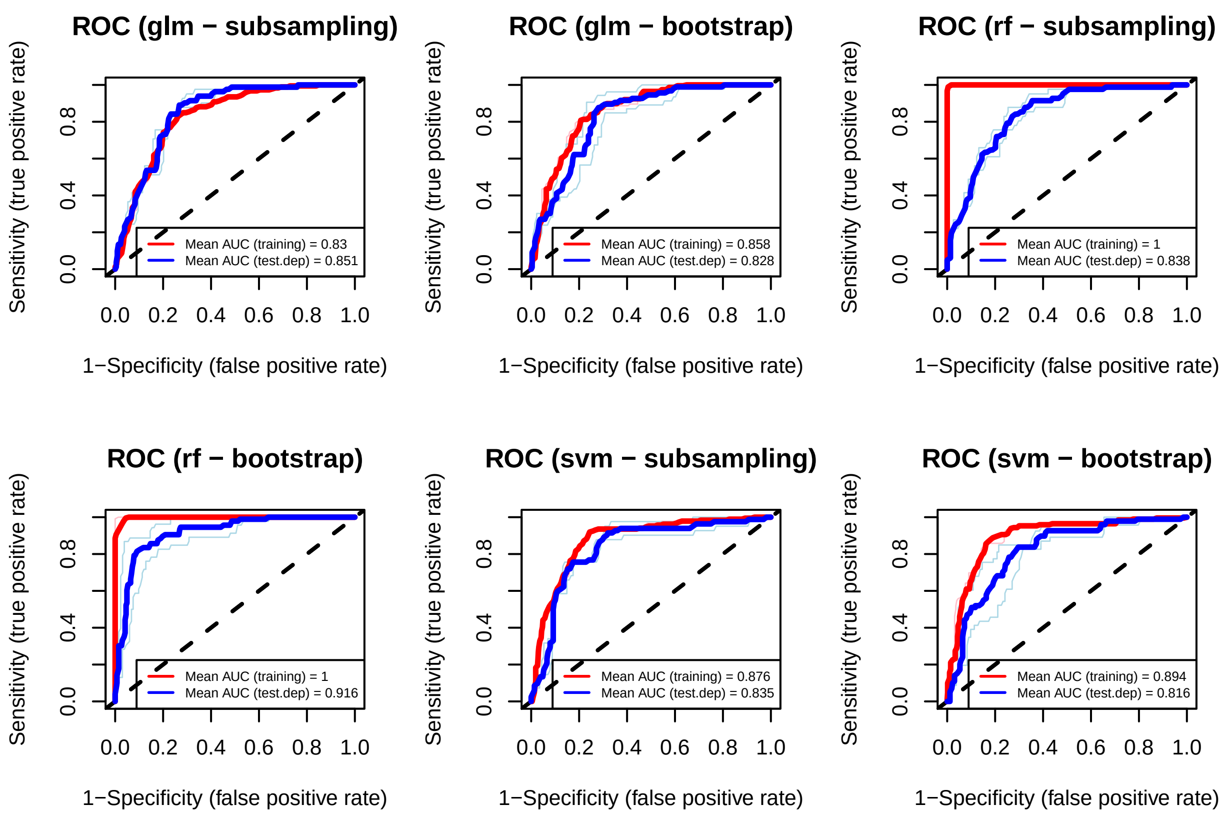


You can use more advanced procedures for resampling and repeating the procedure to see the variation. Currently, three resampling procedures are supported by sdm including "subsampling", "cross-validation", and "bootstrapping".

```
# 2 replications for each of subsampling and bootsrapping (4 in total)

m2 <- sdm(Occurrence~.,data=d,methods=c('glm','rf','svm'),replication=c('sub','boot'),
          test.percent=30,n=2)

#------------

m2
```

```
## class                                    : sdmModels
## ===========================================================
## number of species                        :  1
## number of modelling methods              :  3
## names of modelling methods               :  glm, rf, svm
## replicate.methods (data partitioning)    :  subsampling,bootstrap
## number of replicates (each method)       :  2
## toral number of replicates per model  :  4 (per species)
## test percentage (in subsampling)      :  30
## ------------------------------------------
## model run success percentage (per species)  :
## ------------------------------------------
## method          Occurrence
## ----------------------
## glm        :         100   %
## rf         :         100   %
## svm        :         100   %
##
## ###################################################################
## model Mean performance (per species), using test dataset (generated using partitioning):
## -------------------------------------------------------------------------------
##
##  ## species   :  Occurrence
## =========================
##
## methods    :     AUC     |     COR     |     TSS     |     Deviance
## -------------------------------------------------------------------------
## glm        :     0.84    |     0.52    |     0.62    |     0.86
## rf         :     0.88    |     0.61    |     0.66    |     0.76
## svm        :     0.83    |     0.48    |     0.6     |     1.16
```

```r
roc(m2)
```



run the following code to see the results in a graphical user interface:

```r
gui(m2)
```

You can simply replace preds (climatic variables for the current time), the variables with the SAME NAME but for the future time to get the projection for the future climate scenarios.

Prediction

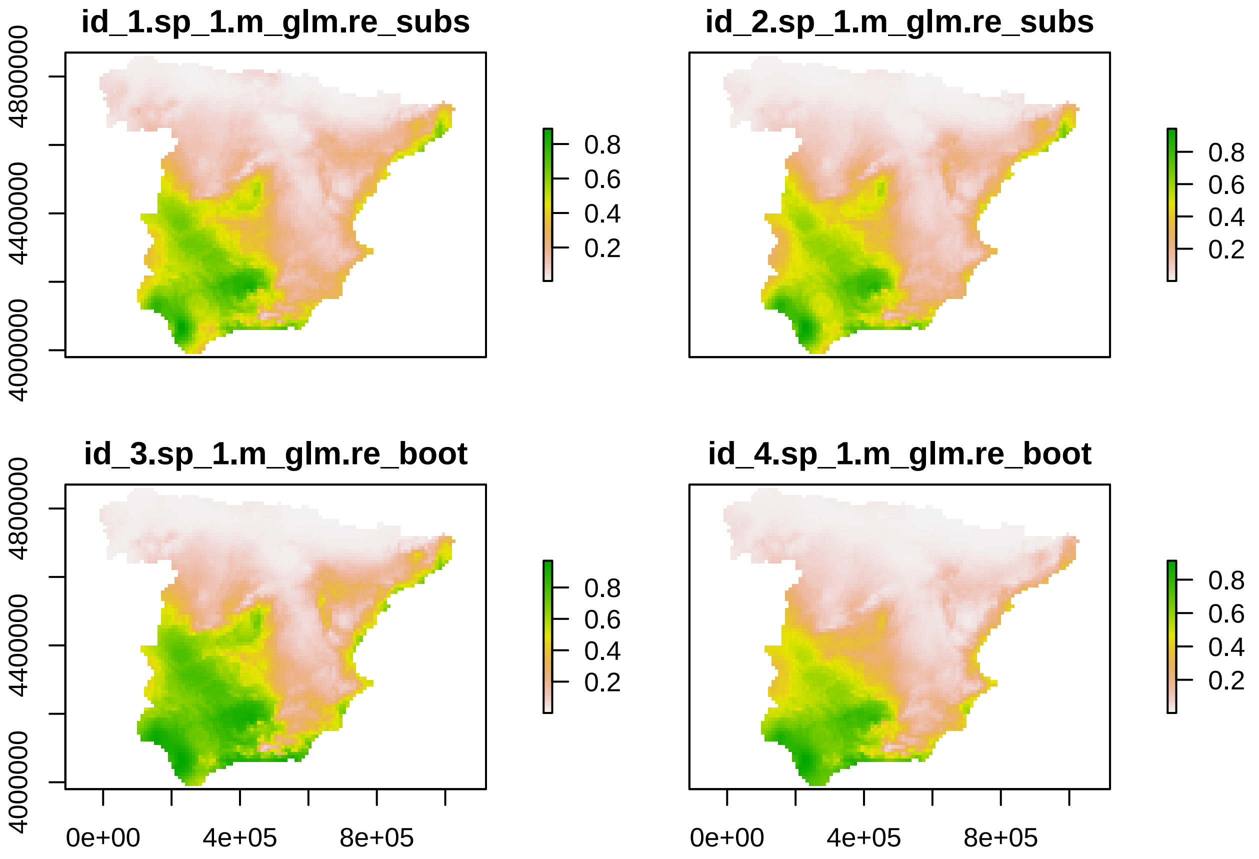
The fitted model object can be used in the predict function to predict the values of probability of occurrence at any (new) location or for a certiain time subject to have the predictor variables for those locations/times.

The prediction is generated for all the methods used to fit the models. For example, in the m2 object, we used 3 modelling methods, 2 replication methods, and 2 runs for each replication, so the predict function generates 3 * 2 * 2 = 12 outputs (in this case RasterLayers). You can limit the function to only use a subset of models or replications, or ask it to take an average over different runs of a method.

```
# in the following, we predict the habitat suitabilities over whole study area
# since the newdata is a raster object, the output is also a raster object

p <- predict(m2,newdata=preds,filename='prediction1.img')

p
```

```
## class       : RasterBrick
## dimensions  : 89, 105, 9345, 12  (nrow, ncol, ncell, nlayers)
## resolution  : 10000, 10000  (x, y)
## extent      : -20625, 1029375, 3979936, 4869936  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=30 +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/babak/Dropbox/_Documents/_course_materials/UVA_Global Ecology and Biodiversity_
## names       : id_1.sp_1.m_glm.re_subs, id_2.sp_1.m_glm.re_subs, id_3.sp_1.m_glm.re_boot, id_4.sp_1.m
## fullname    : id_1-species_Occurrence-method_glm-replication_subsampling, id_2-species_Occurrence-me
```

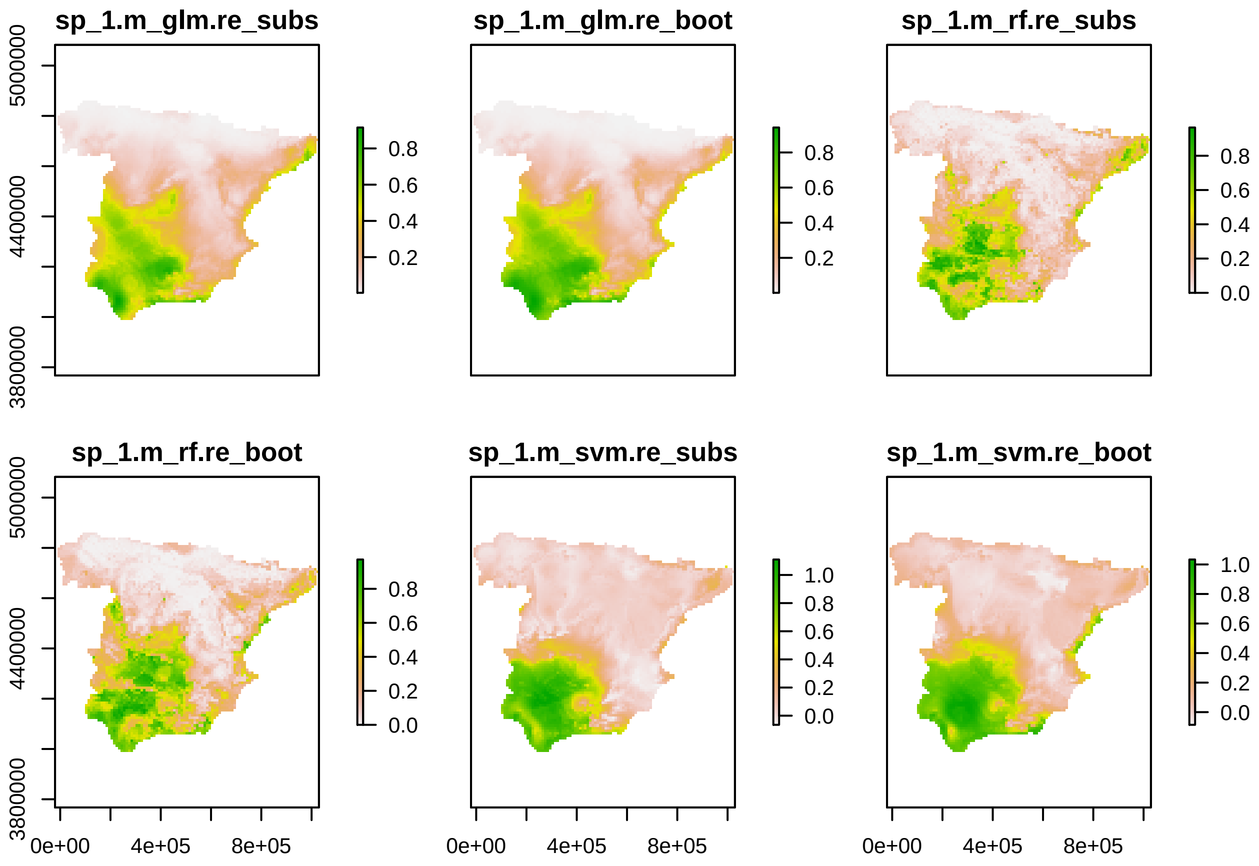
```
plot(p[[1:4]]) # only the firt 4 outputs
```



```
# in the folliwng, for each method, a mean over different replications is returned:
p2 <- predict(m2,newdata=preds,filename='prediction2.img',mean=T)

p2
```

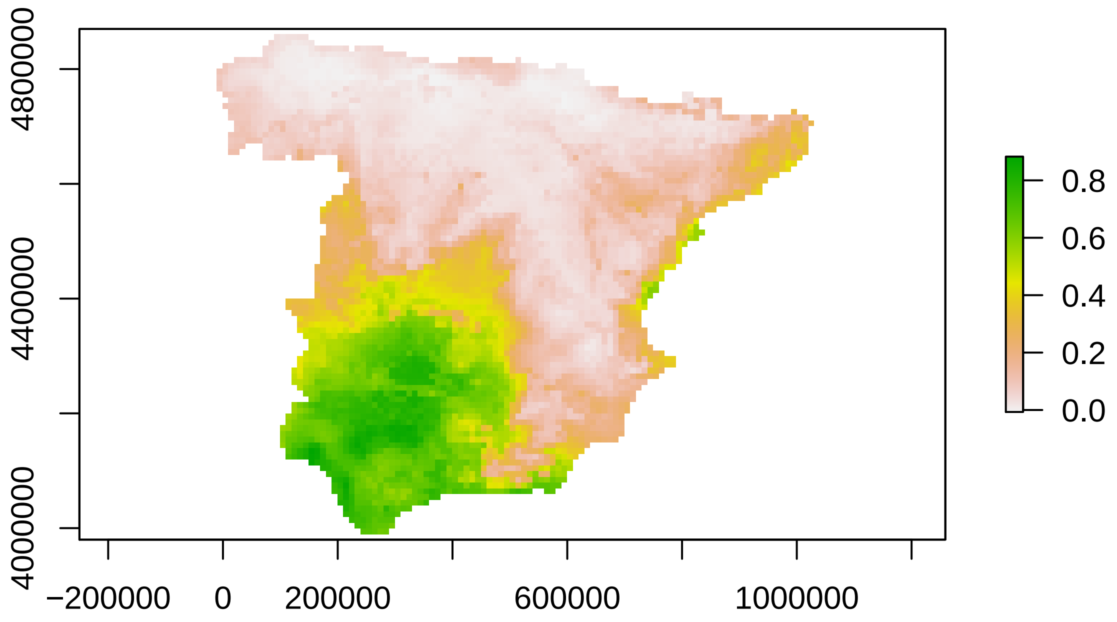
```
## class       : RasterBrick
## dimensions  : 89, 105, 9345, 6  (nrow, ncol, ncell, nlayers)
## resolution  : 10000, 10000  (x, y)
## extent      : -20625, 1029375, 3979936, 4869936  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=30 +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/babak/Dropbox/_Documents/_course_materials/UVA_Global Ecology and Biodiversity_
## names       : sp_1.m_glm.re_subs, sp_1.m_glm.re_boot, sp_1.m_rf.re_subs, sp_1.m_rf.re_boot, sp_1.m_s
## min values  :       3.221634e-03,       9.174619e-04,     -2.640388e-16,     -1.594280e-16,      -6.
## max values  :          0.9155057,          0.9419702,         0.9648750,         0.9724167,
## fullname    : species_Occurrence-method_glm-replication (Mean)_subsampling, species_Occurrence-metho
```

```
plot(p2)
```



To generate an ensemble forecasting based on combining all the models, the ensemble function should be used:

```
# ensemble using a wighted averaging method for which TSS is used. TSS is a threshold-dependent:

# based on the option 2 (opt=2) which specifies threshold should be selected at the
# value that maximizes TSS, is introduced:

ens <- ensemble(m2,newdata = preds,filename='ens1.grd',
                setting=list(method='weighted',stat='TSS',opt=2))

plot(ens)
```


4800000
4400000
4000000
−200000
0
200000
600000
1000000
0.8
0.6
0.4
0.2
0.0

Assignment 1: Select a species that is of your interest. Use the gbif function in the package dismo to download the species (hint: in the gbif function, make sure you included two arguments: geo=TRUE, sp=TRUE; the data will be downloaded as a SpatialPointsDataFrame). The records are presence-only. Then, use the getData function from the raster package to download the 19 bioclim variables. Use both the vifcor and vifstep functions for testing the multicollinearity issue. Which variables are suggested to be excluded by both the functions? What is the difference between the results of these two functions and why?

Assignment 2: Make sure that the species data downloaded from GBIF is a SpatialPoints-DataFrame with only a column in its attribute table that keeps the species occurrence values (since it is a presence-only dataset, all should be 1). Then use the remained predictors (after excluding the collinear variables) together with the species data to generate the sdmdata object. Since it is a presence-only dataset, you need to generate pseudo-absences (background). So, make sure that you add the appropriate argument to the sdmData function for generating the background data. In the end, how many records are available in the sdmdata object? Is the number of presence records is the same as the number of records downloaded from GBIF? If not, what is the reason for the difference?

Assignment 3: Select several modelling methods (examples: glm, brt, rf, bioclim, svm) and fit the sdm models using appropriate settings (i.e., use at least one resampling method that partitions the data into train and test). When the models are fitted, use the gui function to explore the results. Which methods perform the best and worst? Which variable(s) are the most important variables? Is there any useless variable (based on the performance) that you can remove from the model?

Assignment 4: Predict the probability of occurrence as a map for each modelling method. Compare the maps for the best and worst method when you added the species points into the map. Do the maps follow the expected distribution (i.e., the areas that species points are located)? Generate a map using the ensemble function. Does the ensemble map shows any visual improvement?

Assignment 5: Download the bioclilm data for the year 2070 (worst case scenario; i.e., rcp85). Project the probability of occurrence into that time for each individual model, also using the ensemble of all the models. Do you recognise any changes between the current and future times visually?

Challenging Assignment 6: Calculate the changes numerically between the maps generated by the ensemble function (ensemble map) for the current and future times. What is the suitability gain vs loss for the species in future?

Challenging Assignment 7: Convert the ensemble maps of probability of occurrence to presence-absence (using the threshold that maximizes Spe+Sen) for both the current and future times. Then calculate the changes of Presence-Absence values between the two times. What is the ratio between the area of extinction and colonisation for the species in 2070? What proportion of the suitable area persist suitable in the year 2070?

Challenging Assignment 8: Calculate a raster map showing standard deviation of the probability of occurrence values predicted by the different models in the current time. How do you interprete this map?