ECOCHANGE
Biodiversity and ecosystem changes in Europe
http://www.ecochange-project.eu

BIOMOD : Tutorial

Biomod Team :
Wilfried Thuiller, Damien Georges,
Robin Engler, Bruno Lafourcade

February 13, 2012

Contents

1 Before Starting...

In order to facilitate the learning of BIOMOD, a tutorial is provided here with artificial data. It is recommended that the user follows each step and run the models on these artificial datasets, or at least in parallel with runs on its own data. The completion of the tutorial should bring sufficient answers as for the usage of BIOMOD on other datasets.

1.1 Installation and dependencies

To run BIOMOD, please use the latest version of R. A certain number of libraries are also required (rpart, MASS, gbm, gam, nnet, mda, randomForest, Hmisc, plyr) and are also to be downloaded from Rcran before attempting to run BIOMOD.
Note that BIOMOD now enables to build projections directly on rasters. This recent innovation requires several more packages, even if you will not be using rasters with your own work. These are : foreign, sp, rgdal, raster, maptools, some of which are on Rcran and others on the R-forge website.

R code

```
# get list of packages already instaled on my computer
myPackages <- .packages(all = TRUE)
biomodDependencies <- c('rpart','MASS', 'gbm', 'gam', 'nnet', 'mda',
                         'randomForest', 'Hmisc', 'plyr', 'foreign',
                         'sp', 'rgdal', 'raster', 'maptools')
# compare my packages whith those required by BIOMOD
missingPackages <- biomodDependencies[!(biomodDependencies
                                        %in% myPackages)]
# uncomment install.packages() and choose a miror to install
# missing components
if(length(missingPackages) > 0){
  cat(toString(missingPackages), ' packages have to be installed\n')
  #install.packages(missingPackages, dependencies=T)
 } else{ cat('All dependences are installed\n')}
```

R code

```
All dependences are installed
```

BIOMOD is a developping R package that is to be downloaded from the project web page (`https://r-forge.r-project.org/R/?group_id=302`) and install manualy or typing in R :

R code

```
# is biomod already installed
if(!('BIOMOD' %in% myPackages)){
   install.packages("BIOMOD", repos="http://R-Forge.R-project.org")
 } else { cat('BIOMOD is already installed')}
```

R code

```
BIOMOD is already installed
```

It is advised to check relatively frequently for updates.

R code

```
# update biomod if necessary
update.packages("BIOMOD", repos="http://R-Forge.R-project.org")
```

1.2 General adivise

The recommended procedure is to first create a working directory, for example called BIOMOD. Then, create a new folder where to store the datasets, run the models and save the outputs and results. In our examples, we will create and use the directory called Biomod runs. It is from this folder that the files will be read and written. You need to put a copy of your datasets in order to be able to open them once the working directory in R is set to this workspace.

If you want to pause and continue work on this tutoriel (or your own project) later. Just save your session. You will get back all your working space just loading the created file.

R code

```
# save all the working space
save.image("BiomodTutorial.RData")
# free the working space
rm(list=ls())
# and get it back
load("BiomodTutorial.RData")
```

Do keep in mind that some information is kept in the file that has just been generated but that a lot of our work is also stored in the directories that have been created by BIOMOD. Both will be needed for carrying on the next steps.

1.3 Biomod Contents

1.3.1 Biomod Functions

The first thing to do is to load the BIOMOD package. It will load all the functions required to run BIOMOD as well as the examples files to be used in this practical.

R code

```
# load BIOMOD package
library(BIOMOD)
```

R code

```
Loaded gbm 1.6-3.1
```

To access all BIOMOD functions :

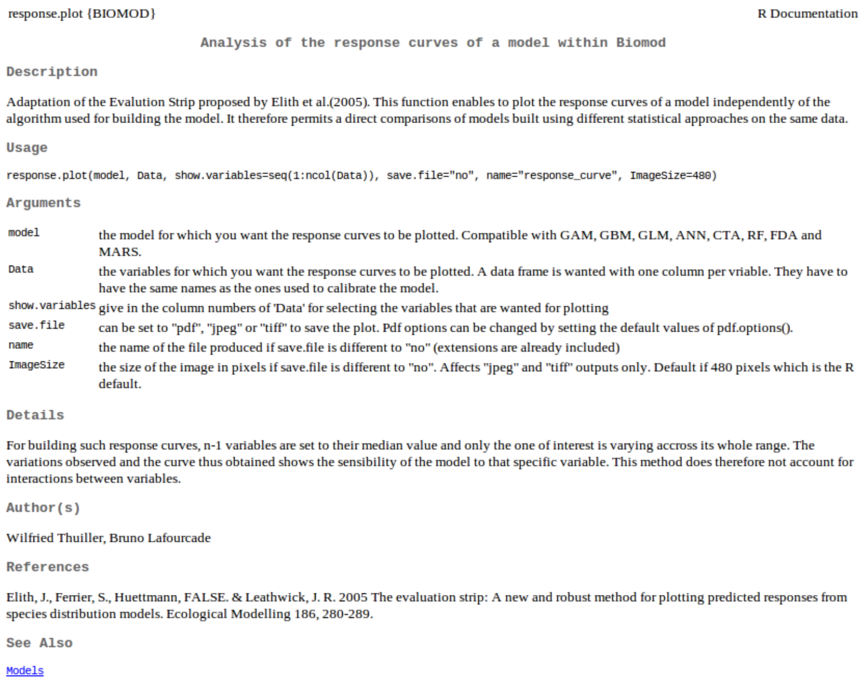
R code

```
# listing of BIOMOD functions
help(package='BIOMOD')
```

As for any function, you can access help files :

R code

```
?response.plot
```



response.plot {BIOMOD} R Documentation

Analysis of the response curves of a model within Biomod

Description

Adaptation of the Evalution Strip proposed by Elith et al.(2005). This function enables to plot the response curves of a model independently of the algorithm used for building the model. It therefore permits a direct comparisons of models built using different statistical approaches on the same data.

Usage

```
response.plot(model, Data, show.variables=seq(1:ncol(Data)), save.file="no", name="response_curve", ImageSize=480)
```

Arguments

`model`	the model for which you want the response curves to be plotted. Compatible with GAM, GBM, GLM, ANN, CTA, RF, FDA and MARS.
`Data`	the variables for which you want the response curves to be plotted. A data frame is wanted with one column per vriable. They have to have the same names as the ones used to calibrate the model.
`show.variables`	give in the column numbers of 'Data' for selecting the variables that are wanted for plotting
`save.file`	can be set to "pdf", "jpeg" or "tiff" to save the plot. Pdf options can be changed by setting the default values of pdf.options().
`name`	the name of the file produced if save.file is different to "no" (extensions are already included)
`ImageSize`	the size of the image in pixels if save.file is different to "no". Affects "jpeg" and "tiff" outputs only. Default if 480 pixels which is the R default.

Details

For building such response curves, n-1 variables are set to their median value and only the one of interest is varying accross its whole range. The variations observed and the curve thus obtained shows the sensibility of the model to that specific variable. This method does therefore not account for interactions between variables.

Author(s)

Wilfried Thuiller, Bruno Lafourcade

References

Elith, J., Ferrier, S., Huettmann, FALSE. & Leathwick, J. R. 2005 The evaluation strip: A new and robust method for plotting predicted responses from species distribution models. Ecological Modelling 186, 280-289.

See Also

Models

You can also open the Biomod pdfs directly from R :

R code

```
#to open the old but detailed version
Biomod.Manual()
#to open one of the latest versions : several pdf files
Biomod.Manual("Biomod_Presentation_Manual")
```

BIOMOD is composed of a series of functions that enables to do our species modelling :

- **Running BIOMOD**
  - Initial.State
  - Models
  - Projection
  - Ensemble.Forecasting
- **Further BIOMOD steps**
  - CurrentPred
  - PredictionBestModel
  - ProjectionBestModel
  - Biomod.Turnover
  - Biomod.RangeSize
  - Migration
- **Plotting functions**
  - level.plot
  - multiple.plot
  - response.plot
- **Other functions**
  - ProbDensFunc : *calculates density probabilities*
  - pseudo.abs : *generating pseudo-absences*
  - BiomodManual : *opens the pdf manual and practicals from R*

We will mainly focus here on the *Models* function as it contains all the options for calibrating and evaluating the models and look at how it can lead to significant variablity in prediction making. This function runs the models and evaluation technics presented in the Presentation Manual of BIOMOD (see *Biomod.Manual('Presentation')*).

1.3.2 Biomod dataset

We need to import the species and the environmental data for our modelling. In our example the same file holds the two datasets.

R code

```
# Loading the example datasets
# For practical reasons, species and environment datasets
# are stored together
data(Sp.Env)
head(Sp.Env)
```

R code

```
  Idw      X     Y   Var1 Var2  Var3  Var4  Var5  Var6
1  73 -9.288 38.62 0.6683 4296 770.1 39.33 295.1 16.74
2  74 -9.292 39.52 0.7596 4174 928.1 57.32 348.7 16.41
3  75 -9.290 39.07 0.7424 4173 870.3 50.05 330.0 16.41
4  76 -8.715 37.72 0.5543 4264 620.0 24.99 239.1 16.66
5  77 -8.717 37.27 0.5489 4169 622.3 25.16 241.0 16.40
6  78 -8.148 37.72 0.5363 4206 591.8 25.74 222.9 16.49
   Var7 Sp281 Sp290 Sp277 Sp164 Sp163 Sp177 Sp185 Sp191
1 10.87     0     1     0     0     1     0     0     1
2 10.51     0     1     0     0     1     0     0     1
3 10.50     0     0     0     0     1     0     0     1
4 10.93     0     0     0     0     0     0     0     0
5 11.28     0     0     0     0     0     0     0     0
6 10.13     0     0     0     0     0     0     0     0
```

- **Idw:** An Id to keep track of the row numbers
- **X and Y:** longitude and latitude of our sites (for plots, not needed for the modelling in itself)
- **Var1 to Var7 :** Environmental variables (bioclimatic in that case)
- **Sp281 to Sp191:** Presence/absence of 8 species

To avoid, confusion, we will split the dataset into 3 part :

- the points coordinates (*LatLong*)
- the bioclimatic data (*Expl.Var*)
- the species occurences(*Resp.Var*)

R code

```
#Visualisation of our data (show first six rows)
LatLong <- Sp.Env[,2:3] # coordinates of points
Expl.Var <- Sp.Env[,4:10] # bioclimatic variables
Resp.Var <- Sp.Env[,11:17] # species occurences
```

BIOMOD does not read the coordinates and does not recognise any geographical information when proceeding the modelling. The user should ensure that all datasets are kept in the same order, i.e. each species information (presence or absence) is correctly associated to the explanatory variables. Any mismatch will not be recognised by BIOMOD and the influence on the different outputs and results will be unnoticeable but real.

To load your own data from a text file, use the *read.table()* function:

R code

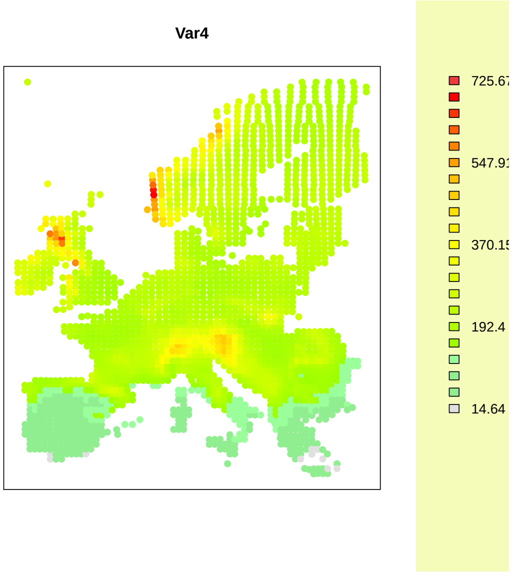
```
#Loading from a text file
#My.Data <- read.table("my_data.txt", h=T, sep="\t")
```

1.3.3 Ploting the data

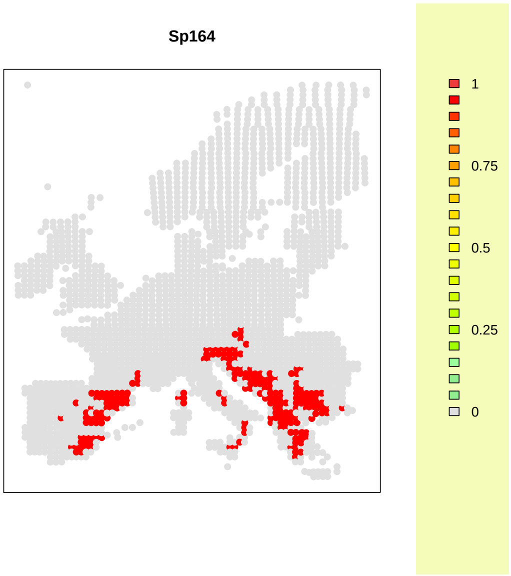
The *level.plot* function requires two inputs : the vector of values that you want to plot and the coordinates of your data points. It works with any type of data.

R code

```
level.plot(Expl.Var[,4], LatLong[,1:2], title=colnames(Expl.Var)[4])
```

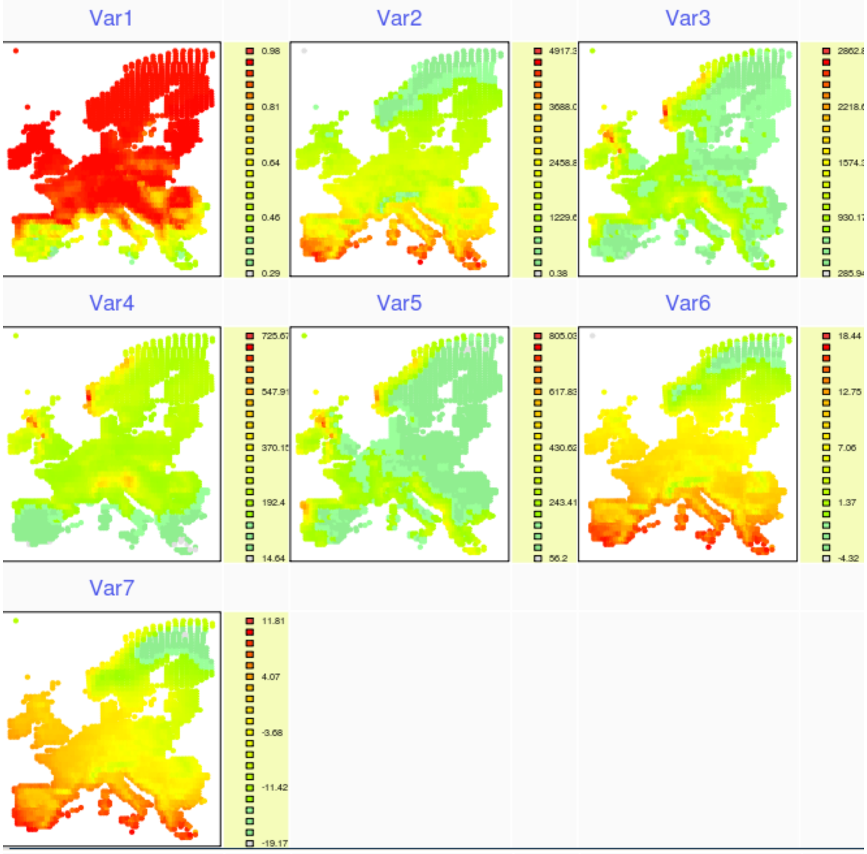


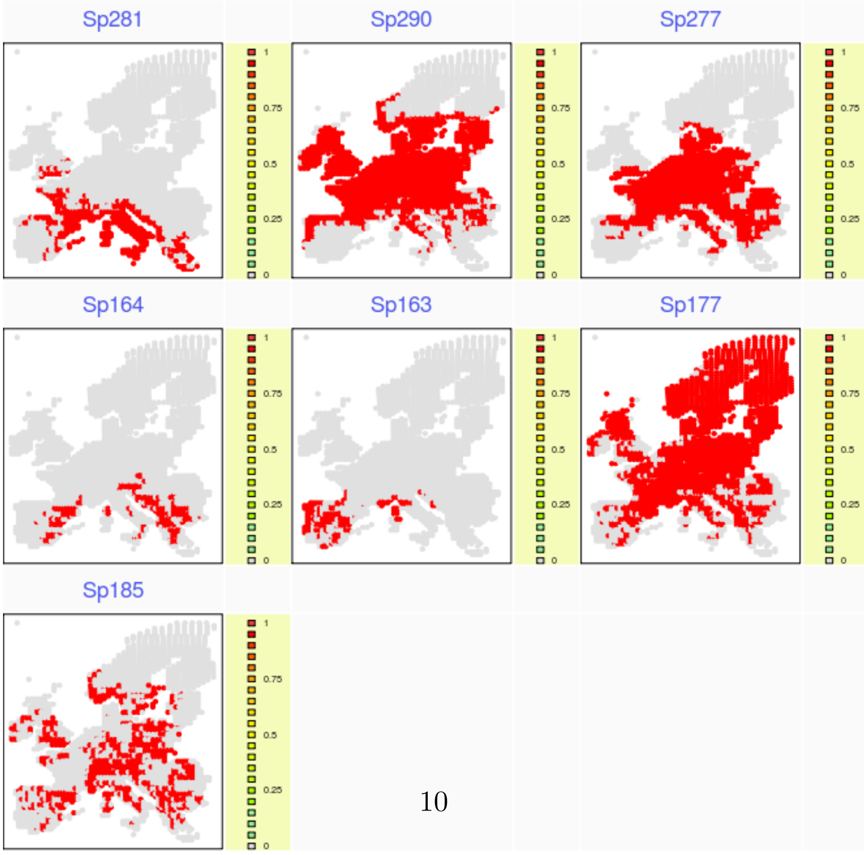
R code
```
level.plot(Resp.Var[,4], LatLong[,1:2], title=colnames(Resp.Var)[4])
```



Let's take a general view of our data with the multiple.plot function :

R code
```
multiple.plot(Expl.Var, LatLong[,1:2], cex=0.7)
multiple.plot(Resp.Var, LatLong[,1:2], cex=0.7)
```


Var1
0.98
0.81
0.64
0.46
0.29
Var2
4917.3
3688.0
2458.8
1229.6
0.38
Var3
2862.8
2218.6
1574.3
930.17
285.94
Var4
725.67
547.91
370.15
192.4
14.64
Var5
805.03
617.83
430.62
243.41
56.2
Var6
18.44
12.75
7.06
1.37
-4.32
Var7
11.81
4.07
-3.68
-11.42
-19.17


Sp281
Sp290
Sp277
Sp164
Sp163
Sp177
Sp185
1
0.75
0.5
0.25
0

You can modify the color gradient by setting the *color.gradient* argument to either *red* (the default), *blue* or *grey*.

2 Initialisation of Biomod

First, we need to set up the dataset in a correct format for BIOMOD by means of the *Initial.State* function. The syntax in the function is the following:

- **Response :** The response variables to model.
- **Explanatory:** The explanatory or independent variables.

Additional arguments (see the *Presentation* pdf for explanation) :

- **IndependentResponse :** Truly independent response variables.
- **IndependentExplanatory :** Truly independent explanatory variables.

These are used to evaluate the predictive accuracy of the models.

We will work on *Sp.Env* dataset, see 1.3.2 to load it correctly or adapt the folowing code lines to fit whith your own data.

So our call looks like :

R code

```
Initial.State(Response = Resp.Var[,1:2], Explanatory = Expl.Var)
```

But we will inform anyway the 2 optional arguments with the same information. The point is to have an example of predictions on our full database as we are going to use pseudo-absences for the purpose of the example (hence BIOMOD will only produce predictions on partial data).

So instead we have :

R code

```
Initial.State(Response = Resp.Var[,1:2],
              Explanatory = Expl.Var,
              IndependentResponse = Resp.Var[,1:2],
              IndependentExplanatory = Expl.Var)
ls()
```

R code

```
[1] "biomodDependencies" "Biomod.material"
[3] "DataBIOMOD"         "DataEvalBIOMOD"
[5] "Expl.Var"           "LatLong"
[7] "missingPackages"    "myPackages"
[9] "Resp.Var"           "Sp.Env"
```

It creates 'DataBIOMOD' our reference database, and DataEvalBIOMOD if you have given independent information. The latter will be used during the testing of the models. Make sure to always keep these datasets unchanged and never delete them.

R code

```
head(DataBIOMOD)
```

R code

```
    Var1 Var2  Var3  Var4  Var5  Var6  Var7 Sp281 Sp290
1 0.6683 4296 770.1 39.33 295.1 16.74 10.87     0     1
2 0.7596 4174 928.1 57.32 348.7 16.41 10.51     0     1
3 0.7424 4173 870.3 50.05 330.0 16.41 10.50     0     0
4 0.5543 4264 620.0 24.99 239.1 16.66 10.93     0     0
5 0.5489 4169 622.3 25.16 241.0 16.40 11.28     0     0
6 0.5363 4206 591.8 25.74 222.9 16.49 10.13     0     0
```

DataBIOMOD contains the environmental variables in the first columns, followed by the species occurrences. DataEvalBIOMOD has the same structure but it contains the data for testing the models.

An object called Biomod.material is also produced which contains information that has been extracted from the datasets like the number of variables, the number of species, etc.. Most of the functions will refer to this object to obtain some necessary values, so make sure to keep it unchanged.

R code

```
Biomod.material
```

R code

```
$NbVar
[1] 7

$VarNames
[1] "Var1" "Var2" "Var3" "Var4" "Var5" "Var6" "Var7"

$NbSpecies
[1] 2

$species.names
[1] "Sp281" "Sp290"
```

3 Settings in Models()

The *Models()* function will run the different models available in BIOMOD and described in the *Presentation* manual. There are two main issues to consider : which models to select and what calibration/evaluation procedure to choose. Let's first have a look at the options to be set in the *Models()* function (arguments are presented with their default values):

```
    Models(
Setting the models to TRUE or FALSE (to run them or not) and their associated options (please refer to the Presentation Manual)
GLM=FALSE, TypeGLM=”simple”, Test=”AIC”,
GBM=FALSE, No.trees= 5000,
GAM=FALSE, Spline=3,
CTA=FALSE, CV.tree=50,
ANN=FALSE, CV.ann=5,
SRE=FALSE, quant=0.025,
FDA=FALSE,
MARS=FALSE,
RF=FALSE,

    The calibration procedure options
NbRunEval=1, DataSplit=100,
NbRepPA=0, strategy=”sre”, coor=NULL, distance=0, nb.absences=NULL,
Yweights=NULL,

    The evaluation procedure options
VarImport=0,
Roc=FALSE, Optimized.Threshold.Roc=FALSE, Kappa=FALSE, TSS=FALSE,
KeepPredIndependent=FALSE
)
```

Note that the various models' specific options will directly influence **the inner** calibration procedure of the models, whereas the calibration options below (NbRunEval, DataSplit) determine **the general trend** of the calibration which will be applied to all the models in the same way.

3.1 Calibration and evaluation procedure

A key issue in modelling is the calibration procedure of the models with the constant effort to obtain a reliable estimation of their performance.

Ideally, one should always evaluate the predictive performance of a model using independent data, i.e. data from which the model didn't obtain any information to build itself. this would enable to reliably test its predictive accuracy on a new dataset and certify its efficiency. Unfortunately, this kind of information is rarely accessible in species distribution modelling. An alternative to assess the predictive performance of the models is to split the original data in calibration (training) and evaluation (testing) datasets : one part is used to feed the model, the other, kept aside and therefore new to the model, is used to check the models' efficiency to predict the right value. As a consequence, this method consists of a trade-off between the amount of data used for the construction of the model and the accuracy of the evaluation measure.

This splitting procedure, widely used in the modelling world, nevertheless brings a major issue : the subsequent randomness of the data selection used for calibration and its impact on the modelling quality.

To obtain a reliable way of evaluating the models while not influencing the prediction making by the random splitting of the data, BIOMOD proposes to built a series of models. The above calibration/evaluation procedure is repeated a certain number of times to perform a reliable evaluation as an attempt to free ourselves from the random effect (the mean result is extracted). Then a final model is built without splitting the data, i.e. 100 % of the data available is used, thus using all the information available and not having any random effect in the prediction making.

This method is also a good way of assessing for uncertainty. While many modellers are satisfied with running only their models once, we propose to build a large number of models to measure the sensitivity of the models to the initial conditions (the input data given). Each model built is kept and can be used to later render projections.

The combination of the two arguments below will determine in which way the models will be built and tested.

- NbRunEval: number of random data splitting procedure for creating calibration and evaluation datasets ; a model will be built from each one of them. If set to zero, only the final 100 % model is built.
- DataSplit: the ratio used for splitting the original database in calibration and evaluation subsets (value to give is the % awarded for calibration). A 70/30 % partitining is recommended as commonly used (Arajo, et al. 2005b, Guisan and Thuiller 2005).

pros : It gives a more robust estimate of the predictive performance of each selected model and it also provides an assessment of the sensitivity of the model to the initial conditions, i.e. to the species distribution data.

cons : it lengthens the modelling time needed to build the models (it can be an exceeding amount of time if not done carefully).

main interest : adds variability in the predictions when several runs are made due to the random effect of selecting the data, i.e. each model is not build using the same information, representing the sensibility of the models on the input data.

Example with the fda and species Sp281

Here is an example of the effect of randomness in the prediction making.

R code

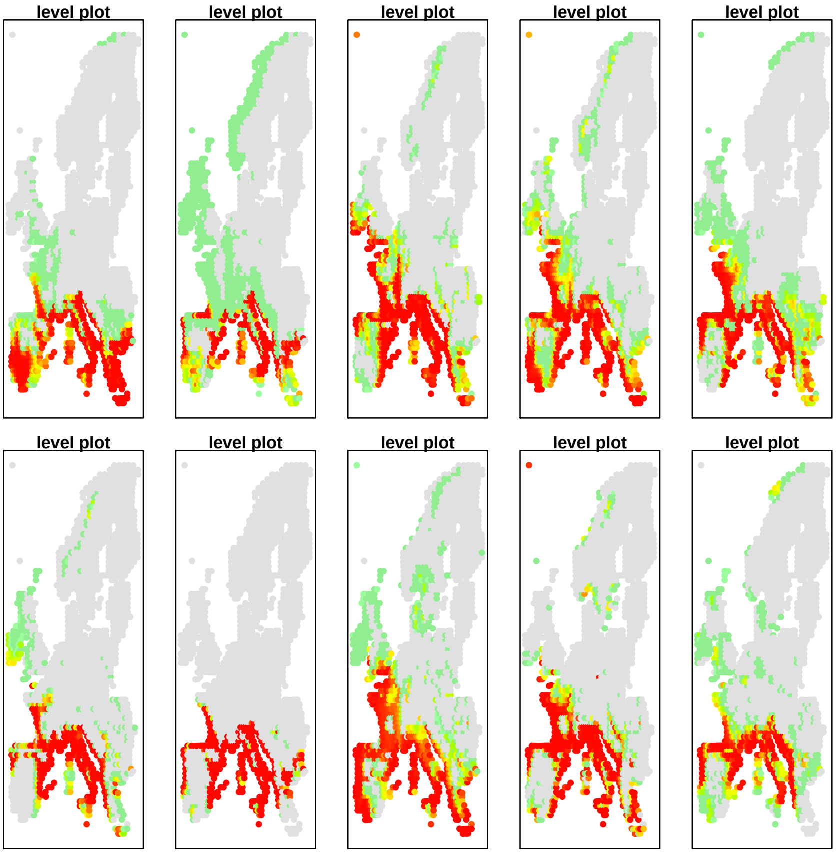
```
#to call our dataset
# library(BIOMOD)
# data(Sp.Env)


store <- matrix(nr=nrow(Resp.Var), nc=0)
for(i in 1:10){
 rand <- sample(nrow(Resp.Var), 100)
 model <- fda("Sp281 ~Var1 + Var2 + Var3 + Var4 + Var5 + Var6 + Var7",
              data=cbind(Expl.Var[rand,],Resp.Var[rand,]), method=mars)
 store <- cbind(store, predict(model, Sp.Env[,4:10], type="post")[,2])
 }
```

R code

```
for(i in 1:10){
 x11()
 par(mar=c(1,1,1,1))
 level.plot(store[,i], LatLong)
 }
```

R code

```
par(mfrow=c(2,5))
par(mar=c(1,1,1,1))
for(i in 1:10) level.plot(store[,i], LatLong, show.scale=F, cex=0.85)
```



This is the same model (FDA) and the same datasets used, only the initial calibration data is changing. The impact on the geographical patterns can clearly be seen.

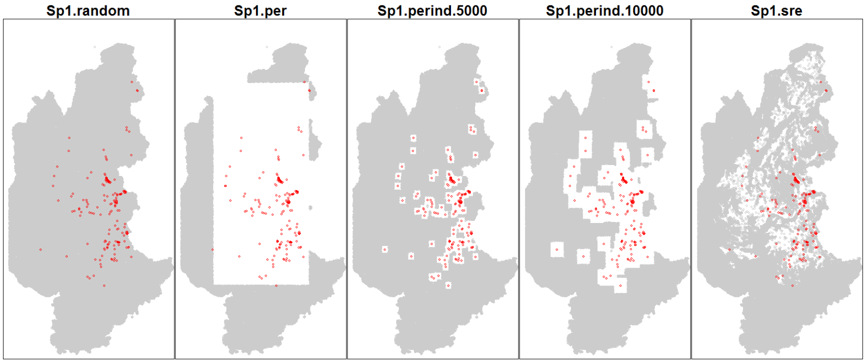
3.2 Pseudo-absences

All the models in BIOMOD need information about presences and absences for being able to determine the suitable conditions for a given species. Some datasets, however, do not contain absences but only presences and the construction of virtual absences is therefore needed. This is, for example, the case of bird datasets where determining an absence can be rather tricky. The assumed absences are called pseudo-absences for there is no field verification of this generated information.
These pseudo-absences are created by considering any point where the species was not recorded and where the environmental conditions are known to cause potential absence. Feeding the models with exceeding numbers of absences can significantly disturb the ability of models to discriminate meaningful relationships between climate and species distributions. Moreover, running

models on such heavy databases is incredibly time consuming.

In addition, some of the chosen absences might unfortunately represent true presences (this is particularly likely in the case of incomplete samples) and therefore the pseudo-absence data gives false information for the estimation of the species-climate relationship. Hence, we propose various strategies that seek to remove the spurious effects of using poorly selected pseudo-absences before running the models.

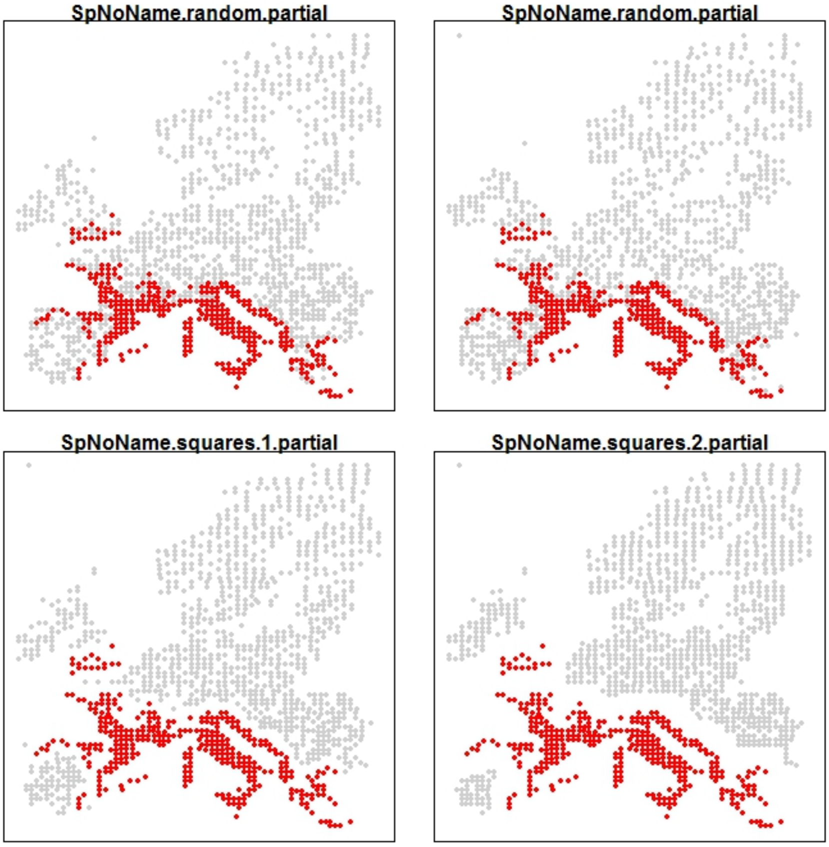
Example of the 4 available strategies in the region of the French Alps for *Larix decidua miller*. The presences are in red and the pseudo-absences selected by each strategy are in grey.



A few examples of what datasets would be created in our case :

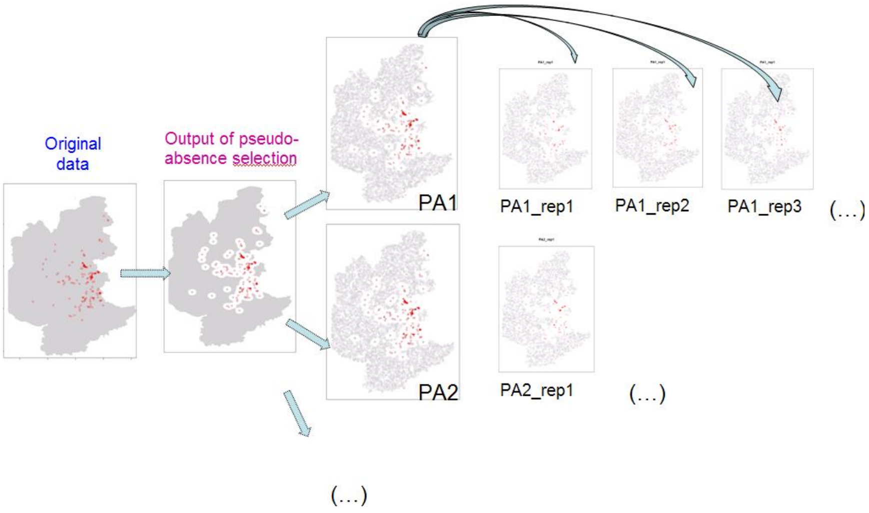
R code

```r
par(mfrow=c(2,2), mar=c(1,1,1,1))
x <- pseudo.abs(LatLong, Resp.Var[,"Sp281"], strategy='random',
                 plot=T, nb.points=1000)
x <- pseudo.abs(LatLong, Resp.Var[,"Sp281"], strategy='random',
                 plot=T, nb.points=1000)
x <- pseudo.abs(LatLong, Resp.Var[,"Sp281"], strategy='squares',
                 plot=T, nb.points=1000, distance=1)
x <- pseudo.abs(LatLong, Resp.Var[,"Sp281"], strategy='squares',
                 plot=T, nb.points=1000, distance=2)
```



In *Models()*, you can choose to run pseudo-absences selections with the argument *NbRepPA*.

This argument is to be correlated with the usage of repetitions for the calibration : once the pool of potential pseudo-absences has been definied by the strategy selected, a user-definied number (Nb.absences argument) is randomly selected from this pool. We therefore have a random effect in the calibration process coming from the creation of pseudo-absences for our data. The NbRepPA argument will define a number of repetitions for randomly withdrawing absences to constitute the calibration datasets. Do consider that the total number of repetitions will be a multiplication of the two repetion arguments



3.3 Weights

The Yweights arguments enables the user to set extra information for the response variables (a matrix with N columns for the N species). This is similar to an index of detectability for each site, which allows users to give stronger weights to more reliable presences or absences. It can be scaled up and put as a weight in the modeling process. For more information, see how *weights* is working in R.

4 Running the models

4.1 Application of Models()

We can now run the different models on our species. It takes only a few moments for each model to run. All the selected models (= TRUE) will run for each species. Here we will have 9(models selected)*4(3 repetitions + final model)*2(PA repetitions) which makes 72 models per species, it will thus take several minutes.

Please, be aware that the *NbRunEval* and *NbRepPA* arguments can considerably enlarge your calculation time by multiplying the number of runs to be made for each species. Do not enter excessively high values for these two arguments **unless** you have sufficient patience and/or reasonable calculation power.

R code

```
Models(GLM = T, TypeGLM = "poly", Test = "AIC",
        GBM = T, No.trees = 2000,
        GAM = T,Spline = 3, CTA = T, CV.tree = 50,
        ANN = T, CV.ann = 2,
        SRE = T, quant=0.025,
        FDA = T,
        MARS = T,
        RF = T,
        NbRunEval = 3, DataSplit = 80, Yweights=NULL,
        Roc = T, Optimized.Threshold.Roc = T, Kappa = T, TSS=T,
        KeepPredIndependent = T, VarImport=5,
        NbRepPA=2, strategy="circles", coor=LatLong,
        distance=2, nb.absences=1000)
```

For the purpose of the example (even though the data does not ask for it) we used 2 pseudo-absences (PA) runs. Note that there has only been one PA run for Sp290 because too little absences were available compared to the ones wanted. The nb.absences argument was set to 1000, but:

R code

```
#the number of data selected by the pseudo-absences procedure
length(Biomod.PA.data$Sp290)
```

R code

```
[1] 1773
```

R code
```
#the number of presences for Sp290
sum(Sp.Env[,"Sp290"])
```

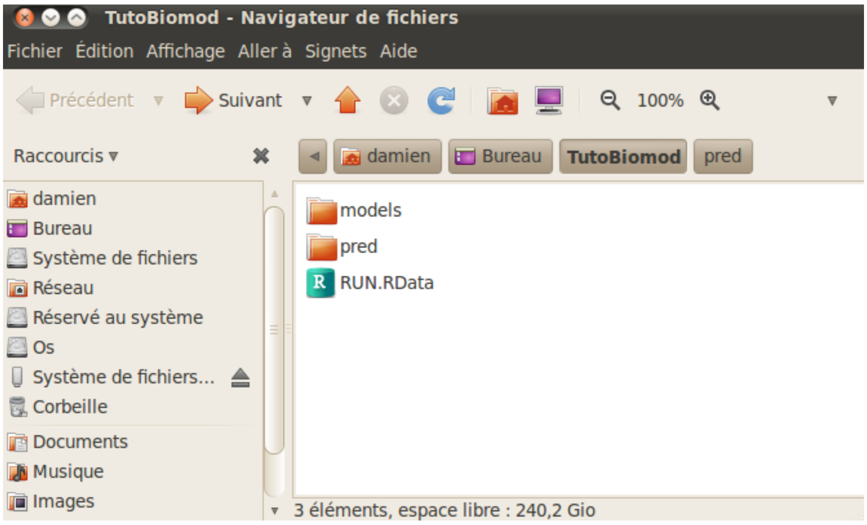
R code
```
[1] 1350
```

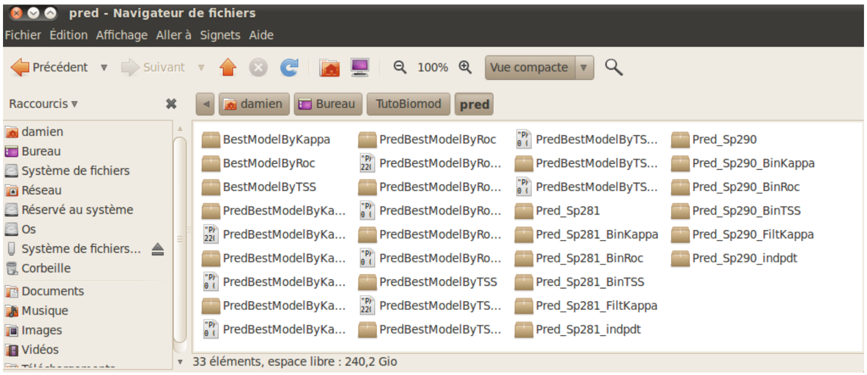
R code
```
#Hence, the number of absences available for calibration
length(Biomod.PA.data$Sp290) - sum(Sp.Env[,"Sp290"])
```

R code
```
[1] 423
```

Too little absences are available. In this case, a single pseudo-absences run is made using all the absences available.

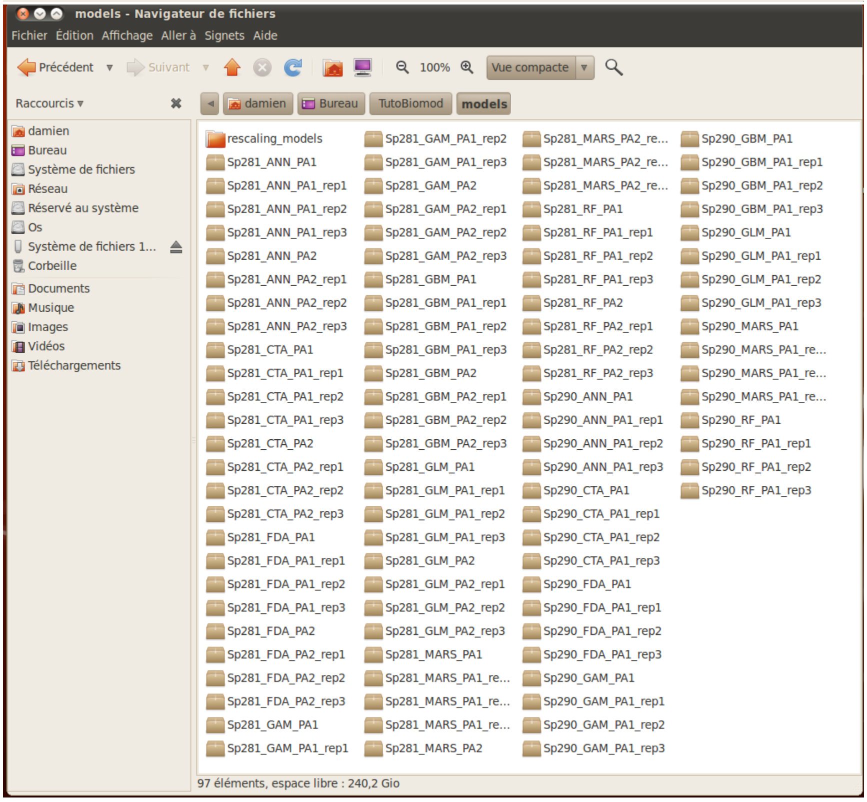
In the latter version of BIOMOD, the results are stored outside R's workspace to counter the memory storage limitations of the software. While running BIOMOD, you will realise that additional folders will be created. A series of objects have been produced in the workspace and also on the hardrive of your computer. Your working folder should now look like this.





BestModelByKappa
BestModelByRoc
BestModelByTSS
PredBestModelByKa...
PredBestModelByKa...
PredBestModelByKa...
PredBestModelByKa...
PredBestModelByKa...
PredBestModelByKa...
PredBestModelByRoc
PredBestModelByRo...
PredBestModelByRo...
PredBestModelByRo...
PredBestModelByRo...
PredBestModelByRo...
PredBestModelByTSS
PredBestModelByTS...
PredBestModelByTS...
PredBestModelByTS...
PredBestModelByTS...
PredBestModelByTS...
Pred_Sp281
Pred_Sp281_BinKappa
Pred_Sp281_BinRoc
Pred_Sp281_BinTSS
Pred_Sp281_FiltKappa
Pred_Sp281_indpdt
Pred_Sp290
Pred_Sp290_BinKappa
Pred_Sp290_BinRoc
Pred_Sp290_BinTSS
Pred_Sp290_FiltKappa
Pred_Sp290_indpdt

33 éléments, espace libre : 240,2 Gio



rescaling_models
Sp281_ANN_PA1
Sp281_ANN_PA1_rep1
Sp281_ANN_PA1_rep2
Sp281_ANN_PA1_rep3
Sp281_ANN_PA2
Sp281_ANN_PA2_rep1
Sp281_ANN_PA2_rep2
Sp281_ANN_PA2_rep3
Sp281_CTA_PA1
Sp281_CTA_PA1_rep1
Sp281_CTA_PA1_rep2
Sp281_CTA_PA1_rep3
Sp281_CTA_PA2
Sp281_CTA_PA2_rep1
Sp281_CTA_PA2_rep2
Sp281_CTA_PA2_rep3
Sp281_FDA_PA1
Sp281_FDA_PA1_rep1
Sp281_FDA_PA1_rep2
Sp281_FDA_PA1_rep3
Sp281_FDA_PA2
Sp281_FDA_PA2_rep1
Sp281_FDA_PA2_rep2
Sp281_FDA_PA2_rep3
Sp281_GAM_PA1
Sp281_GAM_PA1_rep1
Sp281_GAM_PA1_rep2
Sp281_GAM_PA1_rep3
Sp281_GAM_PA2
Sp281_GAM_PA2_rep1
Sp281_GAM_PA2_rep2
Sp281_GAM_PA2_rep3
Sp281_GBM_PA1
Sp281_GBM_PA1_rep1
Sp281_GBM_PA1_rep2
Sp281_GBM_PA1_rep3
Sp281_GBM_PA2
Sp281_GBM_PA2_rep1
Sp281_GBM_PA2_rep2
Sp281_GBM_PA2_rep3
Sp281_GLM_PA1
Sp281_GLM_PA1_rep1
Sp281_GLM_PA1_rep2
Sp281_GLM_PA1_rep3
Sp281_GLM_PA2
Sp281_GLM_PA2_rep1
Sp281_GLM_PA2_rep2
Sp281_GLM_PA2_rep3
Sp281_MARS_PA1
Sp281_MARS_PA1_re...
Sp281_MARS_PA1_re...
Sp281_MARS_PA1_re...
Sp281_MARS_PA2
Sp281_MARS_PA2_re...
Sp281_MARS_PA2_re...
Sp281_MARS_PA2_re...
Sp281_RF_PA1
Sp281_RF_PA1_rep1
Sp281_RF_PA1_rep2
Sp281_RF_PA1_rep3
Sp281_RF_PA2
Sp281_RF_PA2_rep1
Sp281_RF_PA2_rep2
Sp281_RF_PA2_rep3
Sp290_ANN_PA1
Sp290_ANN_PA1_rep1
Sp290_ANN_PA1_rep2
Sp290_ANN_PA1_rep3
Sp290_CTA_PA1
Sp290_CTA_PA1_rep1
Sp290_CTA_PA1_rep2
Sp290_CTA_PA1_rep3
Sp290_FDA_PA1
Sp290_FDA_PA1_rep1
Sp290_FDA_PA1_rep2
Sp290_FDA_PA1_rep3
Sp290_GAM_PA1
Sp290_GAM_PA1_rep1
Sp290_GAM_PA1_rep2
Sp290_GAM_PA1_rep3
Sp290_GBM_PA1
Sp290_GBM_PA1_rep1
Sp290_GBM_PA1_rep2
Sp290_GBM_PA1_rep3
Sp290_GLM_PA1
Sp290_GLM_PA1_rep1
Sp290_GLM_PA1_rep2
Sp290_GLM_PA1_rep3
Sp290_MARS_PA1
Sp290_MARS_PA1_re...
Sp290_MARS_PA1_re...
Sp290_MARS_PA1_re...
Sp290_RF_PA1
Sp290_RF_PA1_rep1
Sp290_RF_PA1_rep2
Sp290_RF_PA1_rep3

97 éléments, espace libre : 240,2 Gio

4.2 Going futher

For those which are interesting in how each model is computed in BIOMOD, you can have a look on the last BIOMOD sumer school 'Methods' practical (*W. Thuiller*). An archived file containing script 'Methods.r' and data required may have been send you with this tutorial.

5 Analysing the outputs

5.1 Objects in the workspace

There are now various objects stored in the workspace. First, we can have a look at what is present in our R session and check what has been produced by the *Models()* function.

R code

```
ls()
```

R code

```
 [1] "BestModelByRoc"            "BestModelByTSS"
 [3] "biomodDependencies"        "Biomod.material"
 [5] "Biomod.PA.data"            "Biomod.PA.sample"
 [7] "DataBIOMOD"                "DataEvalBIOMOD"
 [9] "data.used"                 "Evaluation.results.Kappa"
[11] "Evaluation.results.Roc"    "Evaluation.results.TSS"
[13] "Expl.Var"                  "Expl.Var2"
[15] "Expl.Var3"                 "Future1"
[17] "GBM.list"                  "GBM.perf"
[19] "i"                         "isnullYweights"
[21] "LatLong"                   "missingPackages"
[23] "model"                     "myPackages"
[25] "obj"                       "our.lines"
[27] "Pred"                      "Pred2"
[29] "Pred3"                     "PredBestModelByKappa"
[31] "Pred_Sp281"                "Pred_Sp290"
[33] "Pred_Sp290_BinKappa"       "Pred_Sp290_FiltKappa"
[35] "Pred_Sp290_indpdt"         "rand"
[37] "Resp.Var"                  "Sp290_GLM_PA1"
[39] "Sp290_RF_PA1"              "Sp.Env"
[41] "store"                     "VarImportance"
```

So, we have the outputs generated by *Initial.State* and the original datasets :

- Sp.Env
- LatLong
- Expl.Var
- Resp.Var
- DataBIOMOD
- Biomod.material

We also have the objects produced by the *Models()* function in the workspace (additional objects are stored on the hard disk). These are :

- Evaluation.results.Roc
- Evaluation.results.Kappa
- Evaluation.results.TSS
- VarImportance.

And we get the following if NbRepPA is higher than 0 :

- Biomod.PA.data
- Biomod.PA.sample
- SpNoName.circles.2 (or something close)

5.1.1 Evaluation of the predictive performance

There are three available techniques for making an assessment of a model's performance. A summary table of the type "Evaluation.results.method" are produced containing the predictive performance of each model which is convenient for making comparisons across methods and taxa.

R code

```
#Here we only display the info for the first species modelled
Evaluation.results.Kappa[1:8]
```

R code

```
$Sp281_PA1
     Cross.validation indepdt.data total.score Cutoff
ANN             0.879        0.625      0.9075  438.0
CTA             0.883        0.614      0.9579  630.0
GAM             0.855        0.674      0.8829  629.4
GBM             0.901        0.646      0.9148  592.8
GLM             0.894         0.65      0.8490  699.3
MARS            0.926        0.676      0.9200  429.6
FDA             0.880        0.642      0.9170  105.8
RF              0.930        0.763      1.0000  340.0
SRE             0.658        0.394      0.6675   10.0
     Sensitivity Specificity
ANN        96.17        96.2
CTA        98.98        98.0
GAM        94.13        95.6
```

```
GBM        97.19        96.2
GLM        89.03        95.8
MARS       94.13        97.8
FDA        94.90        97.3
RF        100.00       100.0
SRE        83.42        86.8

$Sp281_PA1_rep1
     Cross.validation indepdt.data total.score Cutoff
ANN             0.841         none      0.8929  171.3
CTA             0.858         none      0.8748  718.5
GAM             0.839         none      0.8561  619.4
GBM             0.896         none      0.9156  654.3
GLM             0.841         none      0.8408  769.2
MARS            0.956         none      0.9281  599.4
FDA             0.853         none      0.8852  109.8
RF              0.930         none      0.9858  410.0
SRE             0.669         none      0.6415   10.0
     Sensitivity Specificity
ANN        99.23        94.0
CTA        98.72        93.1
GAM        93.37        94.3
GBM        95.41        97.0
GLM        85.71        96.8
MARS       92.86        98.8
FDA        92.60        96.4
RF         99.49        99.4
SRE        83.93        84.7

$Sp281_PA1_rep2
     Cross.validation indepdt.data total.score Cutoff
ANN             0.930         none      0.9265  431.6
CTA             0.911         none      0.9403  630.0
GAM             0.855         none      0.8848  609.4
GBM             0.895         none      0.9152  639.2
GLM             0.929         none      0.9278  659.3
MARS            0.900         none      0.9230  239.8
FDA             0.876         none      0.9124  228.6
RF              0.921         none      0.9841  330.0
SRE             0.633         none      0.6936   10.0
     Sensitivity Specificity
ANN        96.94        97.0
CTA        97.45        97.6
GAM        94.39        95.6
GBM        94.64        97.3
GLM        96.17        97.4
MARS       96.68        96.9
FDA        92.60        98.0
RF         99.23        99.4
```

```
SRE        81.63        89.5

$Sp281_PA1_rep3
     Cross.validation indepdt.data total.score Cutoff
ANN             0.866         none      0.8951  395.2
CTA             0.879         none      0.9290  340.0
GAM             0.870         none      0.8946  569.4
GBM             0.913         none      0.9124  612.9
GLM             0.913         none      0.9328  669.3
MARS            0.921         none      0.9243  629.4
FDA             0.911         none      0.9248  302.9
RF              0.938         none      0.9876  420.0
SRE             0.672         none      0.6752   10.0
     Sensitivity Specificity
ANN        94.90        96.0
CTA        98.98        96.3
GAM        96.68        95.2
GBM        95.92        96.6
GLM        95.92        97.8
MARS       92.09        98.9
FDA        93.37        98.4
RF         99.49        99.5
SRE        84.44        86.8

$Sp281_PA2
     Cross.validation indepdt.data total.score Cutoff
ANN             0.868        0.639      0.9493 293.20
CTA             0.868        0.622      0.9309 210.00
GAM             0.848        0.678      0.8859 749.25
GBM             0.903        0.641      0.9235 577.40
GLM             0.763        0.652      0.8437 749.25
MARS            0.921        0.686      0.9345 359.64
FDA             0.916        0.653      0.9118  72.12
RF              0.941        0.767      1.0000 390.00
SRE             0.669        0.394      0.6546  10.00
     Sensitivity Specificity
ANN        98.72        97.6
CTA        99.49        96.2
GAM        88.78        98.1
GBM        97.96        96.4
GLM        86.99        96.4
MARS       95.66        98.0
FDA        94.64        97.1
RF        100.00       100.0
SRE        83.42        85.9

$Sp281_PA2_rep1
     Cross.validation indepdt.data total.score Cutoff
ANN             0.854         none      0.9063  402.0
```

```
CTA             0.860         none     0.9096  630.0
GAM             0.829         none     0.8566  558.9
GBM             0.885         none     0.9089  604.1
GLM             0.758         none     0.7735  625.0
MARS            0.918         none     0.9258  659.3
FDA             0.909         none     0.9226  389.7
RF              0.937         none     0.9876  450.0
SRE             0.714         none     0.6821   10.0
     Sensitivity Specificity
ANN        96.94        95.8
CTA        96.94        96.0
GAM        95.92        93.2
GBM        95.66        96.5
GLM        90.05        90.6
MARS       91.58        99.2
FDA        92.35        98.7
RF         98.98        99.7
SRE        82.65        88.2

$Sp281_PA2_rep2
     Cross.validation indepdt.data total.score Cutoff
ANN             0.865         none      0.8919  591.9
CTA             0.851         none      0.9228  660.0
GAM             0.807         none      0.8390  608.8
GBM             0.886         none      0.9061  657.3
GLM             0.724         none      0.7552  688.6
MARS            0.881         none      0.9268  399.6
FDA             0.884         none      0.9232  179.9
RF              0.913         none      0.9805  380.0
SRE             0.646         none      0.6505   10.0
     Sensitivity Specificity
ANN        97.70        94.6
CTA        96.43        97.0
GAM        92.35        93.7
GBM        93.62        97.2
GLM        85.71        91.5
MARS       93.88        98.3
FDA        93.62        98.2
RF         99.23        99.2
SRE        84.18        85.2

$Sp281_PA2_rep3
     Cross.validation indepdt.data total.score Cutoff
ANN             0.884         none      0.9259  404.0
CTA             0.893         none      0.8865  726.8
GAM             0.908         none      0.8517  598.8
GBM             0.938         none      0.9148  603.8
GLM             0.807         none      0.7467  706.4
MARS            0.965         none      0.9242  329.7
```

```
FDA             0.956         none      0.9219  118.2
RF              0.973         none      0.9947  490.0
SRE             0.646         none      0.6145   10.0
     Sensitivity Specificity
ANN        95.66         97.5
CTA        96.94         94.6
GAM        93.88         93.8
GBM        97.19         96.2
GLM        84.18         91.7
MARS       95.66         97.4
FDA        94.39         97.8
RF         99.49         99.9
SRE        84.18         82.6
```

You can explore and see that the PA2 runs for Sp290 are empty matrices. That's because there has only been 1 PA run for that species.

5.1.2 Evaluation of the importance of each variable

It is always difficult to compare predictions from different models as they do not rely on the same algorithms, techniques and assumptions about the expected relationship between the reponse and the variables, i.e. the species distributions and the environment. With a permutation procedure, BIOMOD proposes another way to examine the importance of the variables in the models. We extract a measure of relative importance of each variable that is independent of the model. Note that the importance of the variables is only calculated for the final model.

Procedure: once the models are trained (i.e. calibrated), a standard prediction is made. Then, one of the variables is randomized and a new prediction is made. The correlation score between that new prediction and the standard prediction is calculated and is considered to give an estimation of the variable importance in the model :

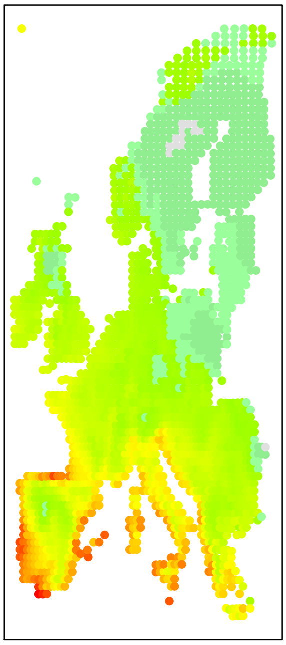
R code

```
model <- glm(Sp281 ~ Var1 + Var2 + Var3 + Var4 + Var5 + Var6 + Var7, data=Sp.Env)
Pred <- predict(model, Expl.Var, type="response")
```

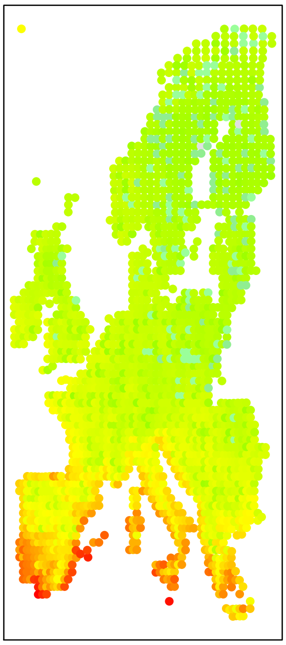
R code

```
Expl.Var2 <- Expl.Var
Expl.Var2[,'Var1'] <- sample(Expl.Var[,'Var1'])
Pred2 <- predict(model, Expl.Var2, type="response")
par(mfrow=c(1,2))
level.plot(Pred, LatLong, show.scale=F, cex=0.8)
level.plot(Pred2, LatLong, show.scale=F, cex=0.8)
```

level plot



level plot



R code

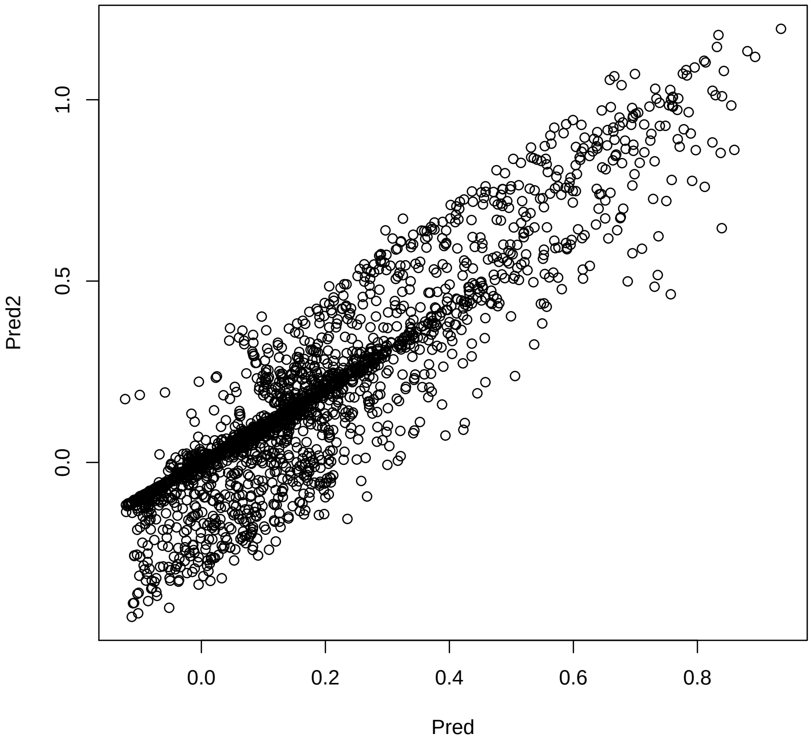
```
cor(Pred, Pred2)
```

R code

```
[1] 0.911
```

R code

```
plot(Pred, Pred2)
```



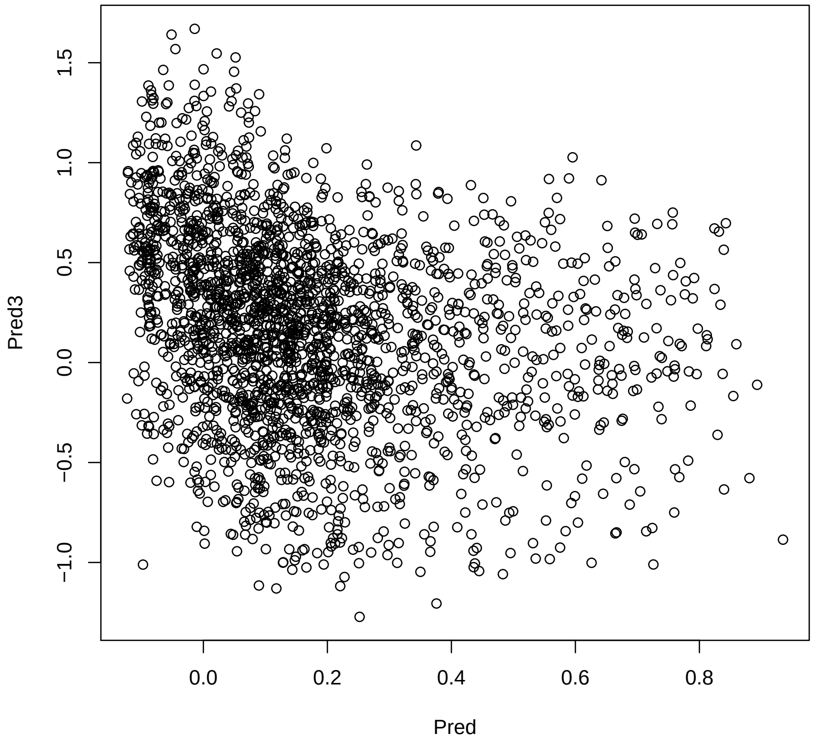
A good correlation score between the two predictions, i.e. they only slightly differ, shows that the randomized variable has little influence on the prediction making and is considered not important for the model in its prediction.

R code

```
Expl.Var3 <- Expl.Var
Expl.Var3[,'Var7'] <- sample(Expl.Var[,'Var7'])
Pred3 <- predict(model, Expl.Var3, type="response")
plot(Pred, Pred3)
cor(Pred, Pred3)
```

R code

```
[1] -0.268
```



In contrary, a low correlation means a significant difference in the prediction making, showing an importance of that variable for the model.

NOTE : in the *VarImportance* output, the values given correspond to 1 minus the correlation score. High values will therefore reveal a high importance of the variable whereas a value close to 0 will reveal no importance.

Score of variable 1 (Pred2) : 1 - cor(Pred, Pred2) = 0.09 meaning low influence
Score of variable 2 (Pred3) : 1 - cor(Pred, Pred3) = 1.27 meaning high influence

This step is repeated n times for each variable independently and the means are kept for each variable.

NOTE : The obtained correlation can be negative. We consider these cases to represent an even bigger influence of the permutated variable on the prediction than with a correlation of 0. The variable importance estimation will therefore still be given as 1 minus the correlation score and, as a consequence, turn into values higher than 1. These cases are not so rare.

Running the *Models* function will produce an object called ”VarImportance” (only if VarImp was put higher than 0 in the function call). The results are stored individually per species and per model. Let’s look at the results we have :

R code

```
VarImportance
```

R code

```
$Sp281
      Var1  Var2  Var3  Var4  Var5  Var6  Var7
ANN  0.279 0.695 0.443 0.478 0.336 0.724 0.972
CTA  0.247 0.143 0.160 0.069 0.168 0.020 0.663
GAM  0.387 1.172 0.618 0.143 0.225 0.415 1.247
GBM  0.142 0.032 0.075 0.035 0.006 0.003 0.600
GLM  0.456 0.067 0.739 0.223 0.202 0.000 0.289
MARS 0.573 0.179 0.062 0.169 0.096 0.000 0.649
FDA  0.364 1.261 0.606 0.258 0.200    NA 1.134
RF   0.154 0.056 0.094 0.075 0.035 0.048 0.423
SRE  0.073 0.039 0.003 0.030 0.062 0.016 0.086

$Sp290
      Var1  Var2  Var3  Var4  Var5  Var6  Var7
ANN  0.000 0.484 0.453 0.364 0.372 0.000 0.447
CTA  0.517 0.210 0.000 0.000 0.000 0.453 0.019
GAM  0.437 0.803 0.000 0.083 0.011 0.285 0.474
GBM  0.180 0.153 0.001 0.070 0.000 0.236 0.004
GLM  0.462 0.649 0.133 0.000 0.077 0.167 0.374
MARS 0.372 0.062 0.000 0.256 0.000 0.596 0.235
FDA  0.380 0.000 0.000 0.078 0.000 0.730 0.050
RF   0.163 0.149 0.015 0.107 0.002 0.208 0.048
SRE  0.018 0.010 0.024 0.013 0.020 0.002 0.042
```

Values should be considered independently for each model. For instance, the SRE shows a generally low value for all the variable when the ANN is generally high. The goal is nevertheless to identify which variable is of the most importance. A good example with the GLM for Sp281, only 2 variables seem to have a significance in the predictions.

Note also that this technic only accounts for the direct effects of the variables and doesn’t enable to identify combined effect of variables or anything as such. It should mainly be considered as an informational tool, not an absolute reliable measure of the variables' contributions to the models.

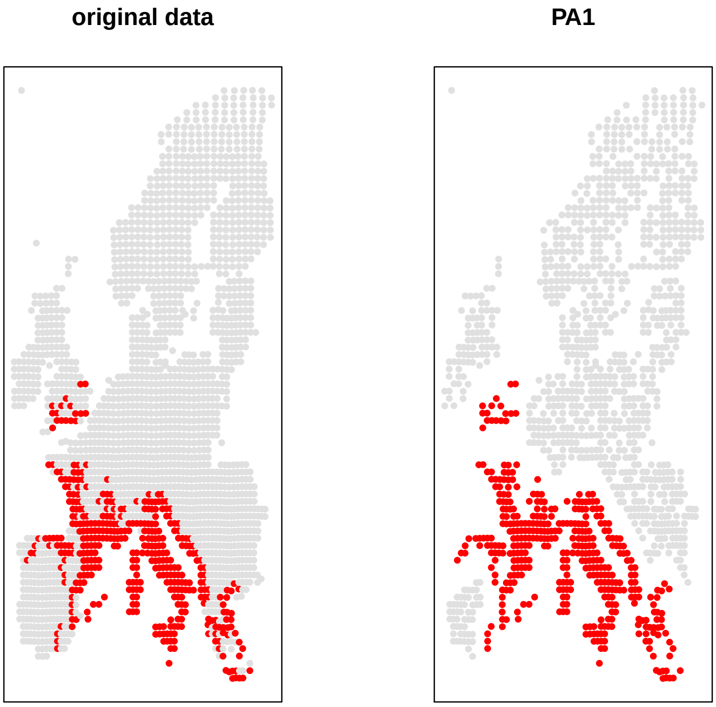
5.1.3 PA data generated

Biomod.PA.data contains the amount of data available after the inner run of the pseudo-absence function. Biomod.PA.sample contains the rows to take from DataBIOMOD to get the data that has been used for the calibration of each species for each PA run.

For example, let's see what data has been used for the calibration of the run PA1 :

R code

```
our.lines <- Biomod.PA.sample$Sp281$PA1
par(mfrow=c(1,2))
level.plot(DataBIOMOD[, "Sp281"], LatLong, title='original data',
            show.scale=F, cex=0.6)
level.plot(DataBIOMOD[our.lines, "Sp281"], LatLong[our.lines,],
            title='PA1', show.scale=F, cex=0.6)
```


original data
PA1

5.2 Objects stored on the hard drive : The Models

Each algorithm (excepted SRE) generates an object storing the different parameterisation, the importance of each variable for the model and other statistics. This output is essential as it allows generating predictions.
These objects, the models themselves, are now stored out of the R workspace directly on the computers' hard disk. They are named after the algorithm used and the species' names, i.e. Sp164_FDA for example. There is also extensions of the names concerning the repetitions and the pseudo-absences runs, so that one of our models will be Sp164_FDA_PA1_rep2.

Back loading the models and having them directly usable is very straightforward : simply use the *load()* function to have the model restored in the R workspace, with the same name plus the directory root. This is also the case with the other outputs stored outside of R (predictions and projections). The syntax is not always handy but easy to pick up :

R code

```
#Example of the GLM
load("models/Sp290_GLM_PA1")
ls()
```

R code

```
 [1] "BestModelByRoc"             "BestModelByTSS"
 [3] "biomodDependencies"         "Biomod.material"
 [5] "Biomod.PA.data"             "Biomod.PA.sample"
 [7] "DataBIOMOD"                 "DataEvalBIOMOD"
 [9] "data.used"                  "Evaluation.results.Kappa"
[11] "Evaluation.results.Roc"     "Evaluation.results.TSS"
[13] "Expl.Var"                   "Expl.Var2"
[15] "Expl.Var3"                  "Future1"
[17] "GBM.list"                   "GBM.perf"
[19] "i"                          "isnullYweights"
[21] "LatLong"                    "missingPackages"
[23] "model"                      "myPackages"
[25] "obj"                        "our.lines"
[27] "Pred"                       "Pred2"
[29] "Pred3"                      "PredBestModelByKappa"
[31] "Pred_Sp281"                 "Pred_Sp290"
[33] "Pred_Sp290_BinKappa"        "Pred_Sp290_FiltKappa"
[35] "Pred_Sp290_indpdt"          "rand"
[37] "Resp.Var"                   "Sp290_GLM_PA1"
[39] "Sp290_RF_PA1"               "Sp.Env"
[41] "store"                      "VarImportance"
```

R code

```
Sp290_GLM_PA1
```

R code

```
Call:  glm(formula = Sp290 ~ poly(Var6, 3) + poly(Var7, 3) + poly(Var2,
    3) + I(Var1^3) + poly(Var5, 3) + poly(Var3, 2), family = binomial,
    data = DataBIOMOD[calib.lines, ], weights = RunWeights[calib.lines])

Coefficients:
   (Intercept)  poly(Var6, 3)1  poly(Var6, 3)2
         -27.5          -138.7           -62.9
poly(Var6, 3)3  poly(Var7, 3)1  poly(Var7, 3)2
         -64.1           141.8          -269.4
poly(Var7, 3)3  poly(Var2, 3)1  poly(Var2, 3)2
         106.6           615.0           -70.7
poly(Var2, 3)3       I(Var1^3)  poly(Var5, 3)1
        -108.7            43.8           -96.8
poly(Var5, 3)2  poly(Var5, 3)3  poly(Var3, 2)1
          66.7            32.1           114.8
poly(Var3, 2)2
         -54.1

Degrees of Freedom: 1772 Total (i.e. Null);  1757 Residual
Null Deviance:          3740
Residual Deviance: 254        AIC: 280
```

R code

```
summary(Sp290_GLM_PA1)
```

R code

```
Call:
glm(formula = Sp290 ~ poly(Var6, 3) + poly(Var7, 3) + poly(Var2,
    3) + I(Var1^3) + poly(Var5, 3) + poly(Var3, 2), family = binomial,
    data = DataBIOMOD[calib.lines, ], weights = RunWeights[calib.lines])

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-3.673   0.000   0.000   0.006   3.572

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)      -27.52       4.03   -6.83  8.8e-12 ***
poly(Var6, 3)1  -138.68     396.79   -0.35  0.72671
poly(Var6, 3)2   -62.87     165.30   -0.38  0.70369
poly(Var6, 3)3   -64.07      39.28   -1.63  0.10289
poly(Var7, 3)1   141.83     134.39    1.06  0.29128
poly(Var7, 3)2  -269.40      47.81   -5.63  1.8e-08 ***
poly(Var7, 3)3   106.58      40.03    2.66  0.00776 **
poly(Var2, 3)1   615.01     315.46    1.95  0.05123 .
poly(Var2, 3)2   -70.70      91.46   -0.77  0.43951
poly(Var2, 3)3  -108.67      32.09   -3.39  0.00071 ***
```

```
I(Var1^3)         43.80      6.19    7.07  1.5e-12 ***
poly(Var5, 3)1   -96.78     28.28   -3.42  0.00062 ***
poly(Var5, 3)2    66.66     25.21    2.64  0.00819 **
poly(Var5, 3)3    32.14     14.23    2.26  0.02387 *
poly(Var3, 2)1   114.81     28.53    4.02  5.7e-05 ***
poly(Var3, 2)2   -54.14     20.52   -2.64  0.00834 **
---
Signif. codes:  0
```

A series of commands enables you to navigate in the object and to extract usefull information from it. Here are a few example that can be used for all algorithms.

R code

```
#simply type its name
Sp290_GLM_PA1
```

R code

```
Call:  glm(formula = Sp290 ~ poly(Var6, 3) + poly(Var7, 3) + poly(Var2,
    3) + I(Var1^3) + poly(Var5, 3) + poly(Var3, 2), family = binomial,
    data = DataBIOMOD[calib.lines, ], weights = RunWeights[calib.lines])

Coefficients:
   (Intercept)  poly(Var6, 3)1  poly(Var6, 3)2
         -27.5          -138.7           -62.9
poly(Var6, 3)3  poly(Var7, 3)1  poly(Var7, 3)2
         -64.1           141.8          -269.4
poly(Var7, 3)3  poly(Var2, 3)1  poly(Var2, 3)2
         106.6           615.0           -70.7
poly(Var2, 3)3       I(Var1^3)  poly(Var5, 3)1
        -108.7            43.8           -96.8
poly(Var5, 3)2  poly(Var5, 3)3  poly(Var3, 2)1
          66.7            32.1           114.8
poly(Var3, 2)2
         -54.1

Degrees of Freedom: 1772 Total (i.e. Null);  1757 Residual
Null Deviance:          3740
Residual Deviance: 254        AIC: 280
```

R code

```
names(Sp290_GLM_PA1)
```

R code

```
 [1] "coefficients"       "residuals"
 [3] "fitted.values"      "effects"
 [5] "R"                 "rank"
 [7] "qr"                "family"
 [9] "linear.predictors" "deviance"
[11] "aic"               "null.deviance"
[13] "iter"              "weights"
[15] "prior.weights"     "df.residual"
[17] "df.null"           "y"
[19] "converged"         "boundary"
[21] "model"             "call"
[23] "formula"           "terms"
[25] "data"              "offset"
[27] "control"           "method"
[29] "contrasts"         "xlevels"
[31] "anova"
```

R code

```
str(Sp290_GLM_PA1)
```

R code

```
List of 31
 $ coefficients     : Named num [1:16] -27.5 -138.7 -62.9 -64.1 141.8 ...
  ..- attr(*, "names")= chr [1:16] "(Intercept)" "poly(Var6, 3)1" "poly(Var6, 3)2" "poly(Va
 $ residuals        : Named num [1:1773] 588.77 1.11 1 1 -1 ...
  ..- attr(*, "names")= chr [1:1773] "1" "2" "16" "17" ...
 $ fitted.values    : Named num [1:1773] 1.70e-03 9.01e-01 1.00 1.00 2.52e-07 ...
  ..- attr(*, "names")= chr [1:1773] "1" "2" "16" "17" ...
 $ effects          : Named num [1:1773] 4.661 -0.229 3.839 -0.824 -1.76 ...
  ..- attr(*, "names")= chr [1:1773] "(Intercept)" "poly(Var6, 3)1" "poly(Var6, 3)2" "poly(V
 $ R                : num [1:16, 1:16] -6.02 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "(Intercept)" "poly(Var6, 3)1" "poly(Var6, 3)2" "poly(Var6, 3)3" ...
  .. ..$ : chr [1:16] "(Intercept)" "poly(Var6, 3)1" "poly(Var6, 3)2" "poly(Var6, 3)3" ...
 $ rank             : int 16
 $ qr               :List of 5
  ..$ qr   : num [1:1773, 1:16] -6.019213 0.04959 0.001657 0.000331 0.000149 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:1773] "1" "2" "16" "17" ...
  .. .. ..$ : chr [1:16] "(Intercept)" "poly(Var6, 3)1" "poly(Var6, 3)2" "poly(Var6, 3)3" ..
  ..$ rank : int 16
  ..$ qraux: num [1:16] 1.01 1.1 1 1 1 ...
  ..$ pivot: int [1:16] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ tol  : num 1e-11
  ..- attr(*, "class")= chr "qr"
 $ family           :List of 12
  ..$ family    : chr "binomial"
  ..$ link      : chr "logit"
  ..$ linkfun   :function (mu)
  ..$ linkinv   :function (eta)
  ..$ variance  :function (mu)
  ..$ dev.resids:function (y, mu, wt)
  ..$ aic       :function (y, n, mu, wt, dev)
  ..$ mu.eta    :function (eta)
  ..$ initialize:  expression({     if (NCOL(y) == 1) {         if (is.factor(y))
  ..$ validmu   :function (mu)
  ..$ valideta  :function (eta)
  ..$ simulate  :function (object, nsim)
  ..- attr(*, "class")= chr "family"
 $ linear.predictors: Named num [1:1773] -6.38 2.21 9.22 12.44 -15.19 ...
  ..- attr(*, "names")= chr [1:1773] "1" "2" "16" "17" ...
 $ deviance         : num 254
 $ aic              : num 280
 $ null.deviance    : num 3743
 $ iter             : int 11
 $ weights          : Named num [1:1773] 1.70e-03 8.91e-02 9.95e-05 3.96e-06 8.05e-07 ...
  ..- attr(*, "names")= chr [1:1773] "1" "2" "16" "17" ...
 $ prior.weights    : Named num [1:1773] 1 1 1 1 3.19 ...
  ..- attr(*, "names")= chr [1:1773] "1" "2" "16" "17" ...
 $ df.residual      : int 1757
 $ df.null          : int 1772
 $ y                : Named num [1:1773] 1 1 1 1 0 0 0 0 0 0 ...
  ..- attr(*, "names")= chr [1:1773] "1" "2" "16" "17" ...
 $ converged        : logi TRUE
 $ boundary         : logi FALSE
 $ model            :'data.frame':        1773 obs. of  8 variables:
  ..$ Sp290        : int [1:1773] 1 1 1 1 0 0 0 0 0 0 ...
  ..$ poly(Var6, 3): poly [1:1773, 1:3] 0.0465 0.0448 0.0417 0.0344 0.0471 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:3] "1" "2" "3"
  .. ..- attr(*, "degree")= int [1:3] 1 2 3
  .. ..- attr(*, "coefs")=List of 2
  .. .. ..$ alpha: num [1:3] 7.9 5.95 8.03
  .. .. ..$ norm2: num [1:5] 1 1773 36104 1347227 37019654
  .. ..- attr(*, "class")= chr [1:2] "poly" "matrix"
  ..$ poly(Var7, 3): poly [1:1773, 1:3] 0.0481 0.0468 0.0436 0.0368 0.0464 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:3] "1" "2" "3"
  .. ..- attr(*, "degree")= int [1:3] 1 2 3
  .. ..- attr(*, "coefs")=List of 2
  .. .. ..$ alpha: num [1:3] -2.5 -5.66 -3.89
  .. .. ..$ norm2: num [1:5] 1.00 1.77e+03 7.73e+04 5.41e+06 2.96e+08
  .. ..- attr(*, "class")= chr [1:2] "poly" "matrix"
  ..$ poly(Var2, 3): poly [1:1773, 1:3] 0.0629 0.0598 0.0544 0.0414 0.064 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:3] "1" "2" "3"
  .. ..- attr(*, "degree")= int [1:3] 1 2 3
  .. ..- attr(*, "coefs")=List of 2
  .. .. ..$ alpha: num [1:3] 1847 2513 2468
  .. .. ..$ norm2: num [1:5] 1.00 1.77e+03 1.52e+09 2.56e+15 3.26e+21
  .. ..- attr(*, "class")= chr [1:2] "poly" "matrix"
  ..$ I(Var1^3)    :Class 'AsIs'  num [1:1773] 0.298 0.438 0.533 0.552 0.123 ...
  ..$ poly(Var5, 3): poly [1:1773, 1:3] 0.02708 0.03962 0.04306 0.04945 0.00827 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:3] "1" "2" "3"
  .. ..- attr(*, "degree")= int [1:3] 1 2 3
  .. ..- attr(*, "coefs")=List of 2
  .. .. ..$ alpha: num [1:3] 179 362 450
  .. .. ..$ norm2: num [1:5] 1.00 1.77e+03 1.83e+07 5.61e+11 2.02e+16
  .. ..- attr(*, "class")= chr [1:2] "poly" "matrix"
  ..$ poly(Var3, 2): poly [1:1773, 1:2] -0.00208 0.00958 0.01441 0.02074 -0.01698 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "1" "2"
  .. ..- attr(*, "degree")= int [1:2] 1 2
  .. ..- attr(*, "coefs")=List of 2
  .. .. ..$ alpha: num [1:2] 798 1470
  .. .. ..$ norm2: num [1:4] 1.00 1.77e+03 1.84e+08 7.29e+13
  .. ..- attr(*, "class")= chr [1:2] "poly" "matrix"
  ..$ (weights)    : num [1:1773] 1 1 1 1 3.19 ...
  ..- attr(*, "terms")=Classes 'terms', 'formula' length 3 Sp290 ~ poly(Var6, 3) + poly(Var
  .. .. ..- attr(*, "variables")= language list(Sp290, poly(Var6, 3), poly(Var7, 3), poly(Va
  .. .. ..- attr(*, "factors")= int [1:7, 1:6] 0 1 0 0 0 0 0 0 0 1 ...
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : chr [1:7] "Sp290" "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var2, 3)" ...
  .. .. .. .. ..$ : chr [1:6] "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var2, 3)" "I(Var1^3)" ..
  .. .. ..- attr(*, "term.labels")= chr [1:6] "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var2, 3)
  .. .. ..- attr(*, "order")= int [1:6] 1 1 1 1 1 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=<environment: 0x451cf18>
  .. .. ..- attr(*, "predvars")= language list(Sp290, poly(Var6, 3, coefs = structure(list(a
  .. .. ..- attr(*, "dataClasses")= Named chr [1:8] "numeric" "nmatrix.3" "nmatrix.3" "nmatr
  .. .. .. ..- attr(*, "names")= chr [1:8] "Sp290" "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var
 $ call             : language glm(formula = Sp290 ~ poly(Var6, 3) + poly(Var7, 3) + poly(Va
 $ formula          :Class 'formula' length 3 Sp290 ~ poly(Var6, 3) + poly(Var7, 3) + poly(V
  .. ..- attr(*, ".Environment")=<environment: 0x451cf18>
 $ terms            :Classes 'terms', 'formula' length 3 Sp290 ~ poly(Var6, 3) + poly(Var7,
  .. ..- attr(*, "variables")= language list(Sp290, poly(Var6, 3), poly(Var7, 3), poly(Var2,
  .. ..- attr(*, "factors")= int [1:7, 1:6] 0 1 0 0 0 0 0 0 0 1 ...
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:7] "Sp290" "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var2, 3)" ...
  .. .. .. ..$ : chr [1:6] "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var2, 3)" "I(Var1^3)" ...
```

```
  .. ..- attr(*, "term.labels")= chr [1:6] "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var2, 3)" "
  .. ..- attr(*, "order")= int [1:6] 1 1 1 1 1 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: 0x451cf18>
  .. ..- attr(*, "predvars")= language list(Sp290, poly(Var6, 3, coefs = structure(list(alp
  .. ..- attr(*, "dataClasses")= Named chr [1:8] "numeric" "nmatrix.3" "nmatrix.3" "nmatrix.
  .. .. ..- attr(*, "names")= chr [1:8] "Sp290" "poly(Var6, 3)" "poly(Var7, 3)" "poly(Var2,
 $ data             :'data.frame':       1773 obs. of  9 variables:
  ..$ Var1 : num [1:1773] 0.668 0.76 0.811 0.82 0.497 ...
  ..$ Var2 : num [1:1773] 4296 4174 3964 3458 4340 ...
  ..$ Var3 : num [1:1773] 770 928 994 1079 568 ...
  ..$ Var4 : num [1:1773] 39.3 57.3 66.9 71.4 24.3 ...
  ..$ Var5 : num [1:1773] 295 349 363 391 215 ...
  ..$ Var6 : num [1:1773] 16.7 16.4 15.8 14.4 16.9 ...
  ..$ Var7 : num [1:1773] 10.87 10.51 9.62 7.72 10.39 ...
  ..$ Sp281: int [1:1773] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ Sp290: int [1:1773] 1 1 1 1 0 0 0 0 0 0 ...
 $ offset           : NULL
 $ control          :List of 3
  ..$ epsilon: num 1e-08
  ..$ maxit  : num 25
  ..$ trace  : logi FALSE
 $ method           : chr "glm.fit"
 $ contrasts        : NULL
 $ xlevels          : Named list()
 $ anova            :Classes
```

R code

```
 #summary
 summary(Sp290_GLM_PA1)
```

R code

```
Call:
glm(formula = Sp290 ~ poly(Var6, 3) + poly(Var7, 3) + poly(Var2,
    3) + I(Var1^3) + poly(Var5, 3) + poly(Var3, 2), family = binomial,
    data = DataBIOMOD[calib.lines, ], weights = RunWeights[calib.lines])

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-3.673   0.000   0.000   0.006   3.572

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)      -27.52       4.03   -6.83  8.8e-12 ***
poly(Var6, 3)1  -138.68     396.79   -0.35  0.72671
poly(Var6, 3)2   -62.87     165.30   -0.38  0.70369
poly(Var6, 3)3   -64.07      39.28   -1.63  0.10289
poly(Var7, 3)1   141.83     134.39    1.06  0.29128
poly(Var7, 3)2  -269.40      47.81   -5.63  1.8e-08 ***
poly(Var7, 3)3   106.58      40.03    2.66  0.00776 **
poly(Var2, 3)1   615.01     315.46    1.95  0.05123 .
poly(Var2, 3)2   -70.70      91.46   -0.77  0.43951
poly(Var2, 3)3  -108.67      32.09   -3.39  0.00071 ***
I(Var1^3)         43.80       6.19    7.07  1.5e-12 ***
poly(Var5, 3)1   -96.78      28.28   -3.42  0.00062 ***
poly(Var5, 3)2    66.66      25.21    2.64  0.00819 **
poly(Var5, 3)3    32.14      14.23    2.26  0.02387 *
poly(Var3, 2)1   114.81      28.53    4.02  5.7e-05 ***
poly(Var3, 2)2   -54.14      20.52   -2.64  0.00834 **
---
Signif. codes:  0
```

It shows the information stored, like the different variables retained in the final model.
The outputs also give the different coefficient values, the degrees of freedom, the residual deviance and the AIC of the final model. Of course, each model's outputs will not give the same information, as it depends on its specificity.

The next call obtains the anova results and the details of the stepwise procedure type. Note that the independent variables are ranked by their AIC importance.

R code

```
Sp290_GLM_PA1$anova
```

R code

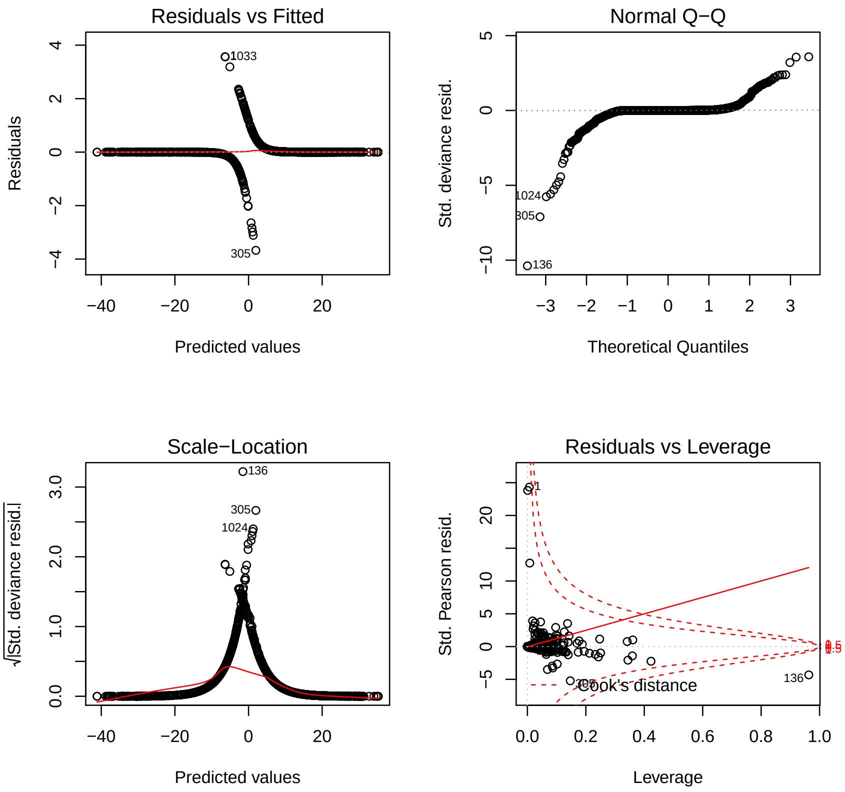
```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
Sp290 ~ 1

Final Model:
Sp290 ~ poly(Var6, 3) + poly(Var7, 3) + poly(Var2, 3) + I(Var1^3) +
    poly(Var5, 3) + poly(Var3, 2)


              Step Df  Deviance Resid. Df Resid. Dev    AIC
1                                    1772     3743.0 3632.7
2  + poly(Var6, 3)  3 2615.2825      1769     1127.7 1099.9
3           + Var1  1  494.2727      1768      633.4  624.1
4  + poly(Var7, 3)  3  285.7301      1765      347.7  355.1
```

```
5  + poly(Var4, 3)  3   23.8282      1762      323.9  337.4
6  + poly(Var2, 3)  3   22.9058      1759      301.0  321.1
7      + I(Var1^3)  1   31.0334      1758      269.9  293.4
8  + poly(Var5, 3)  3    9.5482      1755      260.4  290.2
9  + poly(Var3, 2)  2    7.6884      1753      252.7  286.8
10 - poly(Var4, 3)  3    0.6872      1756      253.4  281.4
11           - Var1  1    1.0252      1757      254.4  280.3
```

The function *plot* of R will give the basic and usual outputs for GLM. They are useful but not entirely relevant in the case of the logistic regression.

R code

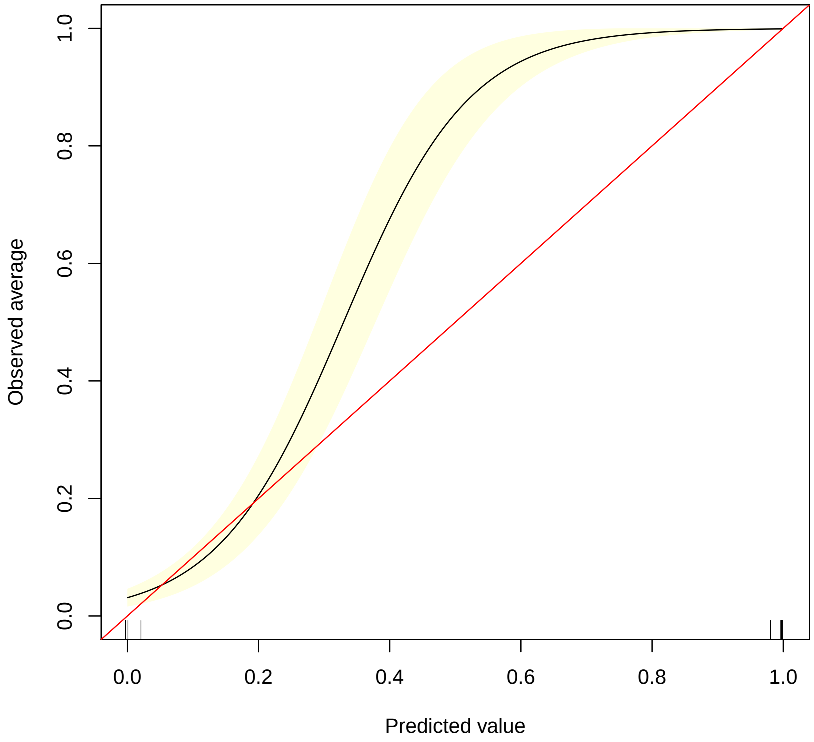
```
par(mfrow=c(2,2))
plot(Sp290_GLM_PA1)
```



The *gbm* library also provides an experimental diagnostic tool that plots the fitted values versus the actual average values. Uses gam to estimate $E(y|p)$. Well-calibrated predictions imply that $E(y|p) = p$. The plot also includes a pointwise 95 band.

This method can be applied to all models to visualise the relative goodness of fit of the model. The function requires the observed presence-absence of the selected species and the predictions. Hence, you will need top load the predictions for this.

R code
```
library(gbm)
load("pred/Pred_Sp290")
#let's store the data that was used for calibration of the
#first PA run for Sp290 to simplify the code
data.used <- DataBIOMOD[Biomod.PA.sample$Sp290$PA1,"Sp290"]
calibrate.plot(data.used, Pred_Sp290[,"GLM",1,1]/1000)
```



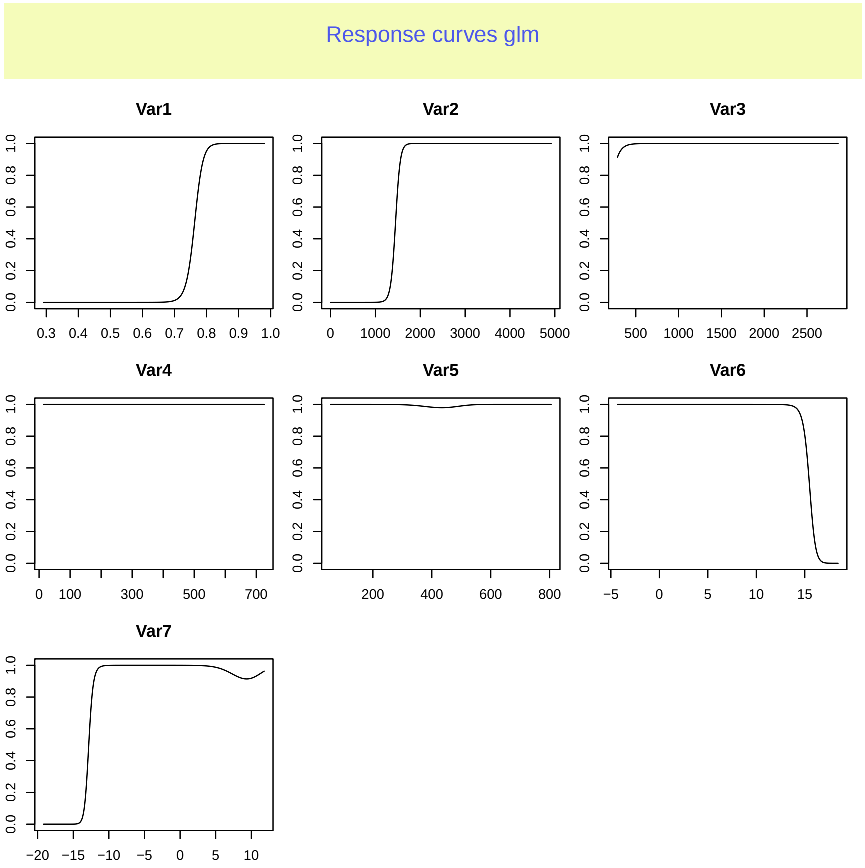
5.2.1 Response Curves

BIOMOD allows plotting the response curves of every model in the good scale. The *response.plot* function must be used to this matter. This function requires a model and a related set of variables to plot the response curves.

Here are two examples of the GLM and RF for the first species modelled. You need to load the model, type its name in the first argument, then give the variables for which you want to see the curves. Note the you can choose to only show some of the variables with the *show.variables* argument.

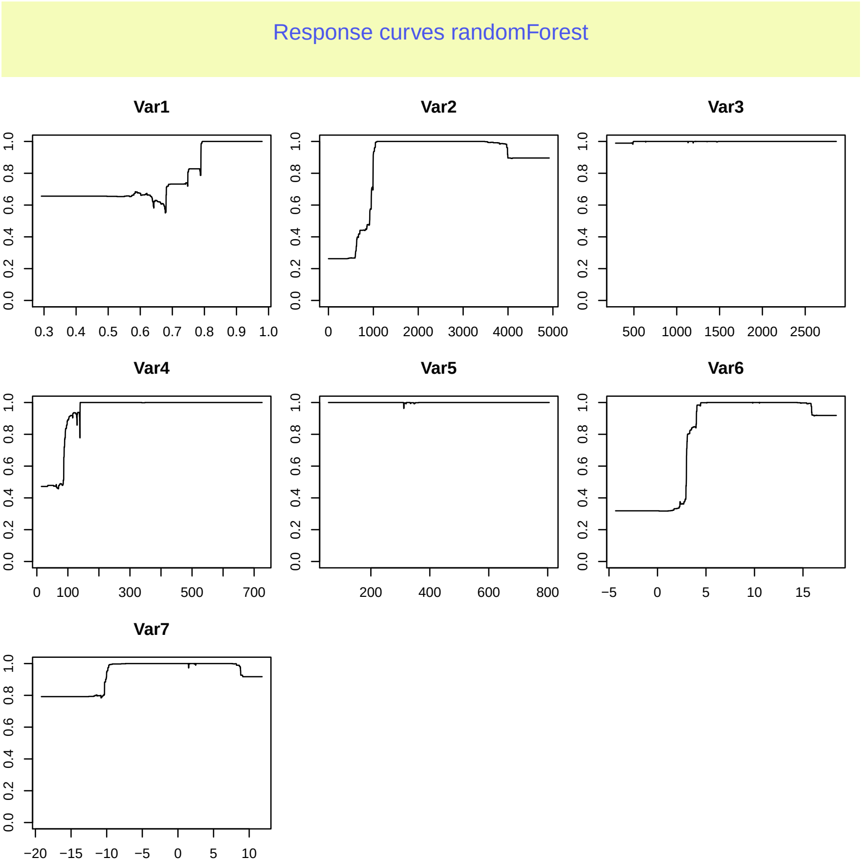
R code

```
#this one has already been loaded in a prior call
response.plot(Sp290_GLM_PA1, Expl.Var)
```



R code

```
load("models/Sp290_RF_PA1")
response.plot(Sp290_RF_PA1, Expl.Var)
```



The response curves are generated following this calculation : N-1 variables are held constant at their mean value whilst the variable of interest contains 100 points varying across the maximum and the minimum of its range. Variation in predictions, made to these 100 cells, only reflects the effects of the one selected variable. Thus, a plot of these predictions allows visualisation of the modelled response to the variable of interest, contingent on the other variables being held constant. This is done subsequently for all the selected variables.

In our examples, the variable Var4 doesn't seem to have a great influence for the GLM (very few variations in the prediction staying close to 1) when it shows a non negligeable influence in the predictions of the RF.

These results are interesting when put together with the VarImportance results. They show that Var5 which shows variability from one model to another doesn't has a high importance for most of the models. In contrast, the variable Var6 which is consistent accross GLM and RF has a big influence on the models. This variable is surely connected with the presence/absence of species 290 and the response plots shows this relationship.

5.3 Objects stored on the hard drive : The Predictions

The predictions made by each model on the original dataset are stored inside the *pred* folder. They are stored independently for each species in an object following a 'Pred.Speciesname' logic and contains the probability of occurrence (habitat suitability index) for each run (if several runs) of the selected models. The same objects are produced for the independent data (if any) and the same logic is respected for the projections.

NOTE: for calculation and memory storage purposes, this index is on a scale between 0 and 1000. To obtain a true probability of occurrence, rescaled between 0 and 1, simply divide each value by a thousand.

R code

```
load("pred/Pred_Sp290")
```

The trick is that these objects are no longer matrices but arrays (multiple dimensions) with 4 dimensions. The dimensions can be visualised as follows :

The first two build up a matrix where each column is the prediction of one of the models. The number of rows corresponds to the amount of data used for building those models.

R code

```
dim(Pred_Sp290)
```

R code

```
[1] 1773    9    4    1
```

R code

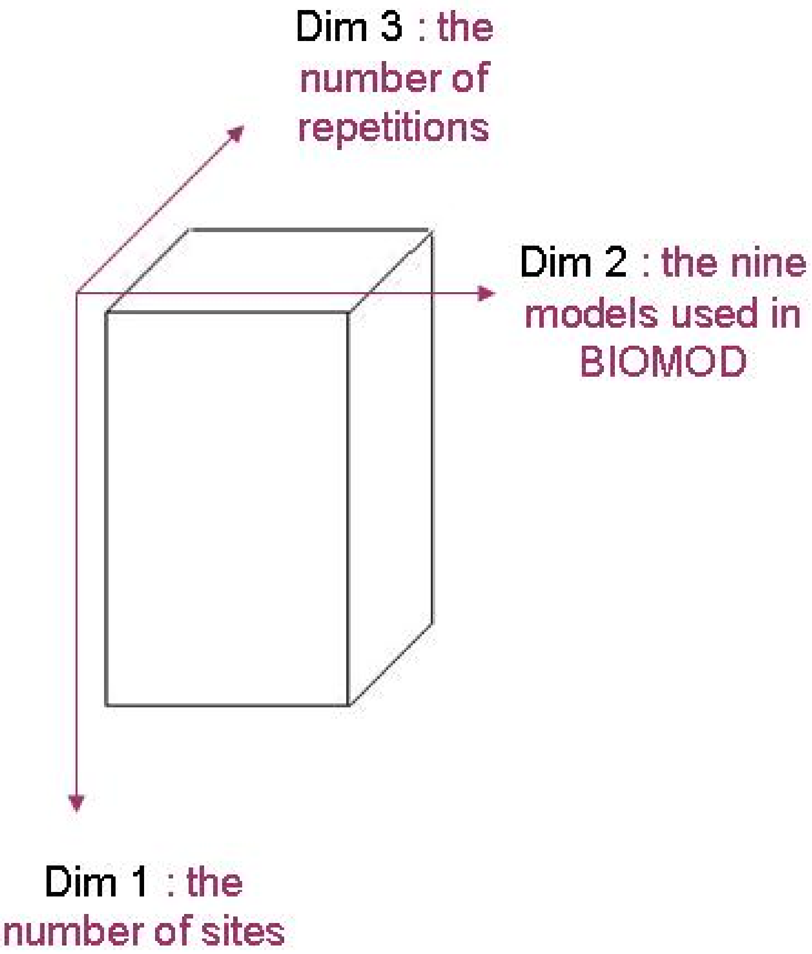
```
Pred_Sp290[1:20,,1,1]
```

R code

```
  ANN CTA GAM GBM GLM MARS FDA  RF SRE
1  61  39 142 412   1   96  35 745   0
2  61 987 777 751 901  974 983 929   0
3  61 987 978 766 999  997 984 988   0
4 986 987 984 766 999  996 984 996   0
5  61  39   1  75   0    3  33   0   0
6  61  39   6  75   0    3  33   0   0
7  61  39   1  75   0    7  33   0   0
8  61  39   9  75   0    3  33   0   0
```

```
9   61  39   1  75   0    9  33    0   0
10  61  39   1  75   0    7  33    0   0
11  61  39  19  75   0    4  33    0   0
12  61  39   1  75   0    9  33    0   0
13  61  39  12  76   0    3  33    0   0
14  61  39  90  79  86    4  33   18   0
15 985 987 995 766 999  998 984  996   0
16 986 987 997 767 999  999 984  997   0
17 989 987 999 906 999  999 984  996   0
18 988 987 999 842 999  999 984  994   0
19 989 987 997 911 999  998 984 1000   0
20 989 987 998 887 999  999 984 1000   0
```

Now, the third dimensions consists of a collection of 2-D matrices, one behind another, corresponding to the prediction produced by each repetition. The minimum for this dimension is 1. Considering that BIOMOD always produces a final model calibrated with 100% of the data given, the length of this third dimension is the value of the NbRunEval argument + 1. For example, with NbRunEval=10, you have 11 layers.


Dim 3 : the number of repetitions
Dim 2 : the nine models used in BIOMOD
Dim 1 : the number of sites

Note that the firts layer is always the final model, then come the repetitions.

R code

```
#the final model
Pred_Sp290[1:15,,1,1]
```

R code

```
   ANN CTA GAM GBM GLM MARS FDA  RF SRE
1   61  39 142 412   1   96  35 745   0
2   61 987 777 751 901  974 983 929   0
3   61 987 978 766 999  997 984 988   0
4  986 987 984 766 999  996 984 996   0
5   61  39   1  75   0    3  33   0   0
6   61  39   6  75   0    3  33   0   0
7   61  39   1  75   0    7  33   0   0
```

```
8    61  39   9  75   0    3  33   0   0
9    61  39   1  75   0    9  33   0   0
10   61  39   1  75   0    7  33   0   0
11   61  39  19  75   0    4  33   0   0
12   61  39   1  75   0    9  33   0   0
13   61  39  12  76   0    3  33   0   0
14   61  39  90  79  86    4  33  18   0
15  985 987 995 766 999  998 984 996   0
```

R code

```
#the first repetition model
Pred_Sp290[1:15,,2,1]
```

R code

```
   ANN CTA GAM GBM GLM MARS FDA  RF SRE
1    31 319 103 376   6  312  33 708   0
2    31 319 598 616 886  990 985 821   0
3    64 319 950 652 999  998 985 934   0
4   580 319 971 653 999  999 985 926   0
5    31  28   0  74   0    2  31   0   0
6    31  28   3  74   0    2  31   1   0
7    31  28   0  74   0    2  31   2   0
8    31  28   5  74   0    2  31   0   0
9    31  28   0  74   0    2  31   1   0
10   31  28   0  74   0    2  31   1   0
11   31  28  14  74   0    2  31   0   0
12   31  28   0  74   0    2  31   0   0
13   31  28   6  75   0    2  31   0   0
14   31  28  91  82  90    3  31  14   0
15  402 319 989 655 999  999 985 952   0
```

R code

```
#the second repetition model
Pred_Sp290[1:15,,3,1]
```

R code

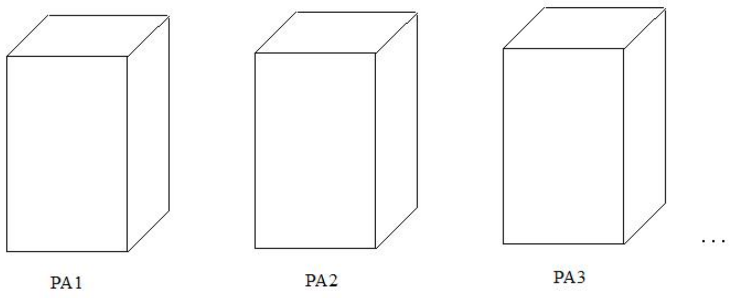
```
   ANN CTA GAM GBM GLM MARS FDA  RF SRE
1    81  36 187 370   0  175  36 410   0
2   925 987 855 757 618  992 983 889   0
3   991 987 988 801 999  998 984 974   0
4   992 987 991 802 999  998 984 984   0
5    45  36   0  74   0    2  32   0   0
6    47  36   1  74   0    4  32   2   0
7    45  36   0  74   0    2  32   0   0
8    47  36   3  74   0    4  32   1   0
```

```
9    46  36   0  74   0    2  32   0   0
10   46  36   0  74   0    2  32   1   0
11   50  36  12  74   0    6  32   0   0
12   47  36   0  74   0    2  32   0   0
13   46  36   3  74   0    5  32   0   0
14   53  36  96  84 178   13  32  29   0
15  997 987 998 802 999  999 984 993   0
```

R code

```
#
#and so on...
```

The fourth dimension represents the number of pseudo-absences repetitions that have been made. In the case where NbRepPA=0, the dimension is simply 1 (not 0).



You will never visualise it this way with R though. It is just an abstract view of how it is sorted. Some usefull functions for not getting lost are *dim()* and *dimnames()*. The first one gives you the number of layers for each dimension, the second will give you their names respectively.

R code

```
load("pred/Pred_Sp281")
dim(Pred_Sp281)
```

R code

```
[1] 1392    9    4    2
```

R code

```
#dimnames(Pred_Sp281)
#you can avoid having the rownames to be printed in the console as they
#are generally not very usefull
dimnames(Pred_Sp281)[-1]
```

R code

```
[[1]]
[1] "ANN"  "CTA"  "GAM"  "GBM"  "GLM"  "MARS" "FDA"  "RF"  
[9] "SRE" 

[[2]]
[1] "total.data" "rep1"       "rep2"       "rep3"      

[[3]]
[1] "PA1" "PA2"
```

For instance, we examine the probability of occurrence of the first species, modelled with CTA. Here we just display 20 rows (or sites) in the middle.

R code

```
 #if you don't inform the 3rd and 4th dimension (you still need commas), you will have all
 #at once in a matrix.
 load("pred/Pred_Sp281")
 Pred_Sp281[281:300,"CTA",,]
```

R code

```
, , PA1

    total.data rep1 rep2 rep3
281       1000  958 1000  997
282        993  958  994  997
283        993  958  994  997
284        993  958  994  997
285        946  836   12  933
286       1000  958 1000  997
287       1000  836 1000  919
288        792  836  792  919
289        993  958  994  997
290        993  958  994  997
291        993  958  994  997
292        993  958  994  629
293        993  958  994  629
294          0  958  910  997
295          0  958    0  997
296          0  958    0    0
297          0  958    0  629
298          0  958    0  629
299        993  958  994    0
300          0  958    0    0

, , PA2
```

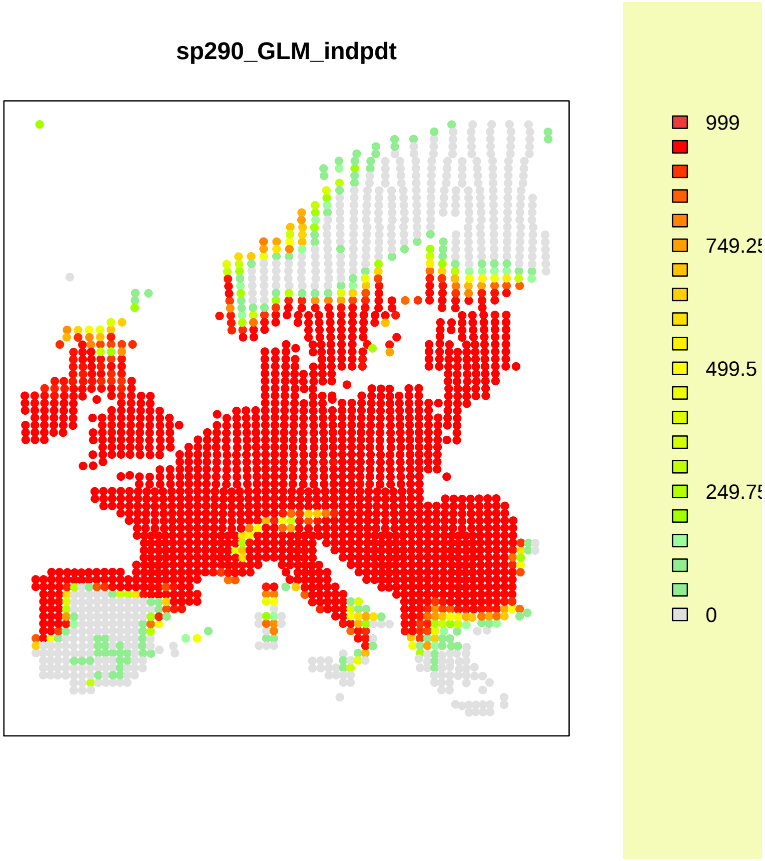
```
    total.data rep1 rep2 rep3
281        967  979  990  969
282        998  979  990  969
283        922 1000  968  827
284        967  976  968  946
285       1000  976  968  946
286       1000  976  968  946
287        800  979  990  969
288        998  979  990  969
289        998  979  990  969
290        800  979  990  969
291        998  979  656   86
292        800  772  990  969
293        800  772  990  969
294        203    0  990  969
295        203    0  990  969
296        967  976  656   86
297        967  976  656   86
298        203  979  990  969
299        203    0  656  809
300        800  979  990  969
```

Note that because there is a random selection of the data for calibration, you will end up with slightly different values on these example runs.

Because we have chosen to run the models with pseudo-absence data, plotting the partial predictions is not very convinient. We will plot instead the values of the fake independent data (which is just the full original dataset) for the GLM.

R code

```
load("pred/Pred_Sp290_indpdt")
level.plot(Pred_Sp290_indpdt[,"GLM",1,1], LatLong, title='sp290_GLM_indpdt', cex=0.8)
```



Note that the independent predictions are only made on the final 100% model and not on the repetitions. To check it :

R code
```
Pred_Sp290_indpdt[1:10,,,]
```

R code
```
, , total.data

   ANN CTA GAM GBM GLM MARS FDA  RF SRE
1   61  39 142 412   1   96  35 745   0
2   61 987 777 751 901  974 983 929   0
3   61 987 669 698 674  946 973 782   0
4   61  39   2  75   0    2  33   6   0
5   61  39   1  75   0    2  33   6   0
6   61  39   2  75   0    2  33   0   0
7   61  39   1  75   0    2  33   0   0
8   61  39  18  79   0    5  33  12   0
9   61  39   7  75   0    3  33   4   0
10  61  39  25  79   0    3  33  10   0
```

```
, , rep1

   ANN CTA GAM GBM GLM MARS FDA RF SRE
1   NA  NA  NA  NA  NA   NA  NA NA  NA
2   NA  NA  NA  NA  NA   NA  NA NA  NA
3   NA  NA  NA  NA  NA   NA  NA NA  NA
4   NA  NA  NA  NA  NA   NA  NA NA  NA
5   NA  NA  NA  NA  NA   NA  NA NA  NA
6   NA  NA  NA  NA  NA   NA  NA NA  NA
7   NA  NA  NA  NA  NA   NA  NA NA  NA
8   NA  NA  NA  NA  NA   NA  NA NA  NA
9   NA  NA  NA  NA  NA   NA  NA NA  NA
10  NA  NA  NA  NA  NA   NA  NA NA  NA

, , rep2

   ANN CTA GAM GBM GLM MARS FDA RF SRE
1   NA  NA  NA  NA  NA   NA  NA NA  NA
2   NA  NA  NA  NA  NA   NA  NA NA  NA
3   NA  NA  NA  NA  NA   NA  NA NA  NA
4   NA  NA  NA  NA  NA   NA  NA NA  NA
5   NA  NA  NA  NA  NA   NA  NA NA  NA
6   NA  NA  NA  NA  NA   NA  NA NA  NA
7   NA  NA  NA  NA  NA   NA  NA NA  NA
8   NA  NA  NA  NA  NA   NA  NA NA  NA
9   NA  NA  NA  NA  NA   NA  NA NA  NA
10  NA  NA  NA  NA  NA   NA  NA NA  NA

, , rep3

   ANN CTA GAM GBM GLM MARS FDA RF SRE
1   NA  NA  NA  NA  NA   NA  NA NA  NA
2   NA  NA  NA  NA  NA   NA  NA NA  NA
3   NA  NA  NA  NA  NA   NA  NA NA  NA
4   NA  NA  NA  NA  NA   NA  NA NA  NA
5   NA  NA  NA  NA  NA   NA  NA NA  NA
6   NA  NA  NA  NA  NA   NA  NA NA  NA
7   NA  NA  NA  NA  NA   NA  NA NA  NA
8   NA  NA  NA  NA  NA   NA  NA NA  NA
9   NA  NA  NA  NA  NA   NA  NA NA  NA
10  NA  NA  NA  NA  NA   NA  NA NA  NA
```

5.3.1 Transforming the predictions on the original dataset

It might be useful to extract the presence/absence predictions. To do so, use the *CurrentPred()* function by switching *BinRoc, BinKappa* and/or *BinTSS* to TRUE and each probability of occurence will be transformed into presence and absence using the cutoff maximising the models accuracy according to Roc, Kappa or TSS. You can also selecting the *FiltRoc, FiltKappa* and/or *FiltTSS* options. That will result in creating a new table where probabilities lower than correspunding optimised cutoff are set to 0 and those upper keep theire value.

R code
```
CurrentPred(GLM=T, GBM=T, GAM=T, CTA=T, ANN=T, SRE=T,
            FDA=T, MARS=F, RF=T, BinRoc=T, BinKappa=T,
            BinTSS=T, FiltKappa=T)
```

New objects are created for each species containing the predictions in binary and or filtered format using the thresholds produced by the evaluation technics : Pred_Sp290_BinRoc, Pred_Sp290_BinKappa, Pred_Sp290_BinTSS, Pred_Sp290_FiltKappa, and so on.

R code
```
load("pred/Pred_Sp290")
load("pred/Pred_Sp290_BinKappa")
load("pred/Pred_Sp290_FiltKappa")
Pred_Sp290[260:270,,1,1]
```

R code
```
    ANN CTA GAM GBM GLM MARS FDA   RF  SRE
260 979 987 998 931 999  998 984 1000    0
261 989 987 987 929 991  997 984 1000 1000
262 979 987 935 932 953  998 984 1000    0
263 989 987 944 930 954  998 984 1000 1000
264 989 987 961 930 954  998 984 1000 1000
265 989 987 999 930 999  997 984 1000 1000
266 989 987 999 930 999  997 984  993 1000
267 989 987 999 930 999  997 984 1000 1000
268 989 987 999 930 999  997 984 1000 1000
269 989 987 999 930 999  997 984  996 1000
270 989 987 998 930 999  997 984 1000 1000
```

R code
```
Pred_Sp290_BinKappa[260:270,,1,1]
```

R code
```
    ANN CTA GAM GBM GLM MARS FDA RF SRE
260   1   1   1   1   1   NA   1  1   0
261   1   1   1   1   1   NA   1  1   1
```

```
262    1    1    1    1    1    NA    1   1    0
263    1    1    1    1    1    NA    1   1    1
264    1    1    1    1    1    NA    1   1    1
265    1    1    1    1    1    NA    1   1    1
266    1    1    1    1    1    NA    1   1    1
267    1    1    1    1    1    NA    1   1    1
268    1    1    1    1    1    NA    1   1    1
269    1    1    1    1    1    NA    1   1    1
270    1    1    1    1    1    NA    1   1    1
```

R code

```
Pred_Sp290_FiltKappa[260:270,,1,1]
```

R code

```
    ANN CTA GAM GBM GLM MARS FDA   RF  SRE
260 979 987 998 931 999   NA 984 1000    0
261 989 987 987 929 991   NA 984 1000 1000
262 979 987 935 932 953   NA 984 1000    0
263 989 987 944 930 954   NA 984 1000 1000
264 989 987 961 930 954   NA 984 1000 1000
265 989 987 999 930 999   NA 984 1000 1000
266 989 987 999 930 999   NA 984  993 1000
267 989 987 999 930 999   NA 984 1000 1000
268 989 987 999 930 999   NA 984 1000 1000
269 989 987 999 930 999   NA 984  996 1000
270 989 987 998 930 999   NA 984 1000 1000
```

R code

```
```

5.3.2 Identifying the best model

In our example, we could compare all the models we run for the different species using the three different evaluation methods available. The function *PredictionBestModel* also transforms the probabilities into the presence/absence and filtered formats.

R code

```
PredictionBestModel(GLM=T,GBM=T, GAM=T, CTA=T, ANN=T, FDA=T,
                    MARS=F, RF=T, SRE=T, method='all',
                    Bin.trans = T, Filt.trans = T)
```

Multimodel comparison according to the TSS statistic:

R code

```
load("pred/BestModelByTSS")
BestModelByTSS
```

R code

```
$Sp281
         Best.Model Cross.validation indepdt.data
PA1              RF            0.947        0.876
PA1_rep1         RF            0.944         none
PA1_rep2        ANN            0.952         none
PA1_rep3         RF            0.955         none
PA2              RF            0.952        0.871
PA2_rep1         RF            0.940         none
PA2_rep2         RF            0.945         none
PA2_rep3         RF            0.972         none
         total.score Cutoff Sensitivity Specificity
PA1           1.0000  340.0      100.00       100.0
PA1_rep1      0.9889  410.0       99.49        99.4
PA1_rep2      0.9394  431.6       96.94        97.0
PA1_rep3      0.9899  420.0       99.49        99.5
PA2           1.0000  390.0      100.00       100.0
PA2_rep1      0.9868  450.0       98.98        99.7
PA2_rep2      0.9843  380.0       99.23        99.2
PA2_rep3      0.9939  490.0       99.49        99.9

$Sp290
         Best.Model Cross.validation indepdt.data
PA1              RF            0.978        0.784
PA1_rep1        GAM            0.981         none
PA1_rep2         RF            0.978         none
PA1_rep3         RF            0.981         none
         total.score Cutoff Sensitivity Specificity
PA1           1.0000  350.0      100.00      100.00
PA1_rep1      0.9666  409.6       98.07       98.58
PA1_rep2      0.9933  710.0       99.33      100.00
PA1_rep3      0.9962  390.0       99.85       99.76
```

The RF comes out first almost each time, let's switch it off : Multimodel comparison according to the TSS statistic:

R code

```
PredictionBestModel(GLM=T,GBM=T, GAM=T, CTA=T, ANN=T, FDA=T,
                     MARS=F, RF=F, SRE=T, method='all',
                     Bin.trans = T, Filt.trans = T)
load("pred/BestModelByTSS")
BestModelByTSS
```

R code

```
$Sp281
         Best.Model Cross.validation indepdt.data
PA1             GBM            0.922        0.776
PA1_rep1        GBM            0.919         none
PA1_rep2        ANN            0.952         none
PA1_rep3        GLM            0.934         none
PA2             GBM            0.918        0.778
PA2_rep1        GBM            0.889         none
PA2_rep2        ANN            0.912         none
PA2_rep3        FDA            0.962         none
         total.score Cutoff Sensitivity Specificity
PA1           0.9388 474.04       98.98        94.9
PA1_rep1      0.9316 568.86       97.96        95.2
PA1_rep2      0.9394 431.64       96.94        97.0
PA1_rep3      0.9446 499.50       97.96        96.5
PA2           0.9436 577.40       97.96        96.4
PA2_rep1      0.9279 527.54       97.19        95.6
PA2_rep2      0.9306 495.00       97.96        95.1
PA2_rep3      0.9222  70.32       95.92        96.3

$Sp290
         Best.Model Cross.validation indepdt.data
PA1             ANN            0.971        0.658
PA1_rep1        ANN            0.981         none
PA1_rep2        GBM            0.970         none
PA1_rep3        GLM            0.974         none
         total.score Cutoff Sensitivity Specificity
PA1           0.9385  358.0       97.63       96.22
PA1_rep1      0.9666  409.6       98.07       98.58
PA1_rep2      0.9732  457.4       98.74       98.58
PA1_rep3      0.9645  459.5       97.63       98.82
```

Multimodel comparison according to the ROC:

R code

```
 load("pred/BestModelByRoc")
 BestModelByRoc
```

R code

```
$Sp281
         Best.Model Cross.validation indepdt.data
PA1             GBM             0.99        0.941
PA1_rep1        GBM            0.988         none
PA1_rep2        GBM            0.991         none
PA1_rep3        GBM            0.992         none
PA2             GBM            0.992        0.941
PA2_rep1        GBM            0.988         none
PA2_rep2        GBM            0.991         none
```

```
PA2_rep3        FDA            0.998        none
         total.score  Cutoff Sensitivity Specificity
PA1            0.996 603.784      96.173        96.3
PA1_rep1       0.994 614.976      96.173        96.2
PA1_rep2       0.993 626.373      96.684        96.7
PA1_rep3       0.995 589.889      96.173        96.2
PA2            0.996 616.454      96.429        96.4
PA2_rep1       0.995 583.706      96.173        96.1
PA2_rep2       0.995 587.232      95.918          96
PA2_rep3       0.987  66.488      96.173        96.1

$Sp290
         Best.Model Cross.validation indepdt.data
PA1             GBM            0.998        0.914
PA1_rep1        GBM            0.998         none
PA1_rep2        GBM            0.997         none
PA1_rep3        GBM            0.998         none
         total.score  Cutoff Sensitivity Specificity
PA1            0.999 450.142      98.593      98.582
PA1_rep1       0.999 460.802      98.593      98.582
PA1_rep2       0.999 476.336      98.593      98.582
PA1_rep3       0.998 408.591          98      98.109
```

.

Multimodel predictions according to the Kappa statistic

R code

```
 load("pred/PredBestModelByKappa")
 PredBestModelByKappa[740:750,,1]
```

R code

```
    PA1 PA1_rep1 PA1_rep2 PA1_rep3 PA2 PA2_rep1 PA2_rep2
740  71       71        0        0  34       23       72
741  71       71        0        0  34       23       72
742  71       71        0        0  34       23       72
743  71       71        0        0  34       23       72
744  71       71        0        0  34       23       72
745  71       71        1        2  34       23       72
746  71       71        1        2  34       23       71
747  71       71        1        0  34       23       75
748  71       71        0        0  34       23       72
749  71       72        1        0  35       23       94
750  71       72        0        0  34       23       72
    PA2_rep3
740       32
741       32
742       32
```

```
743        32
744        32
745        33
746        32
747        33
748        32
749        32
750        32
```

6 Uncertainty analysis

6.1 Models' projection

For all the models currently implemented, BIOMOD is able to project potential distributions of species or land-use classes for other areas, other resolutions or other times. BIOMOD does not utilise the geographical coordinates nor does it perform a re-ordering of the data for making projections. **The user must ensure** that all datasets are kept in the same order in order to allow unmistaken comparisons between observed and predicted maps.

To make the projections, use the function *Projection*.

The syntax is very similar to previous functions. First add the new data (e.g. climate change scenario), then the prefix name of the output (Proj.name), and then the models for which the projections have to be made.

The Proj.name argument is very important as it will be used to store the results and also used by other functions to reload this data. The *Projection* function will create a directory using that name. In our case, it will produce ”proj.Future1” next to ”pred” and ”models” in the working directory. A directory is created for each run of the function with a different scenario.

R code

```
#load the example dataset : future scenario 1
data(Future1)
head(Future1)
Projection(Proj = Future1[,4:10], Proj.name='Future1',
           GLM = T, GBM = T, GAM = T, CTA = T, ANN = T,
           SRE = T, quant=0.025, MARS = T, RF = T,
           BinRoc = T, BinKappa = T, BinTSS = T, FiltRoc = T,
           FiltKappa = T, FiltTSS = T, repetition.models=T)
save.image('RUN.RData')
```

Let's check the future projections made for this scenario :

R code

```
load('RUN.RData')
load("proj.Future1/Proj_Future1_Sp290")
dim(Proj_Future1_Sp290)
```

R code

```
[1] 2264    9    4    1
```

R code

```
dimnames(Proj_Future1_Sp290)[-1]
```

R code

```
[[1]]
[1] "ANN"  "CTA"  "GAM"  "GBM"  "GLM"  "MARS" "FDA"  "RF"  
[9] "SRE" 

[[2]]
[1] "total.data" "rep1"       "rep2"       "rep3"      

[[3]]
[1] "PA1"
```

R code

```
Proj_Future1_Sp290[740:750,,1,1]
```

R code

```
    ANN CTA GAM GBM GLM MARS FDA   RF  SRE
740 989 987 999 930 999  998 984 1000 1000
741 986 987 999 930 999  997 984  992    0
742 989 987 999 930 999  997 984 1000 1000
743 985 987 999 930 999  996 984  985    0
744 979 987 997 929 999  999 984  997 1000
745 981 987 999 930 999  998 984 1000 1000
746 979 916 710 828 853  956 983  956    0
747 989 987 982 930 999  991 984 1000 1000
748 989 987 999 930 999  997 984 1000 1000
749 989 987 999 930 999  998 984 1000 1000
750 989 987 999 930 999  997 984 1000 1000
```

R code

```
load("proj.Future1/Proj_Future1_Sp290_BinRoc")
Proj_Future1_Sp290_BinRoc[740:750,,1,1]
```

R code

```
    ANN CTA GAM GBM GLM MARS FDA RF SRE
740   1   1   1   1   1    1   1  1   1
741   1   1   1   1   1    1   1  1   0
742   1   1   1   1   1    1   1  1   1
743   1   1   1   1   1    1   1  1   0
744   1   1   1   1   1    1   1  1   1
745   1   1   1   1   1    1   1  1   1
746   1   1   1   1   1    1   1  1   0
```

```
747   1   1   1   1   1    1   1  1   1
748   1   1   1   1   1    1   1  1   1
749   1   1   1   1   1    1   1  1   1
750   1   1   1   1   1    1   1  1   1
```

We also have them in binary and filtered format which have been directly produced by the Projection function.

Compare the projections produced with original data

R code

```
multiple.plot(cbind(Sp.Env[,'Sp290'], Proj_Future1_Sp290[,1:8,1,1]), LatLong, cex=0.73)
```

R code

```
load("proj.Future1/Proj_Future1_Sp281")
#PA1 et PA2
multiple.plot(cbind(PA1=Sp.Env[,'Sp281'],PA2=Sp.Env[,'Sp281'], Proj_Future1_Sp281[,1:8,1,1
```

R code

```
#repetitions
multiple.plot(cbind(full=Sp.Env[,'Sp281'], Proj_Future1_Sp281[,1:8,1,2]), LatLong, cex=0.7
multiple.plot(cbind(rep1=Sp.Env[,'Sp281'], Proj_Future1_Sp281[,1:8,2,2]), LatLong, cex=0.7
multiple.plot(cbind(rep2=Sp.Env[,'Sp281'], Proj_Future1_Sp281[,1:8,3,2]), LatLong, cex=0.7
multiple.plot(cbind(rep3=Sp.Env[,'Sp281'], Proj_Future1_Sp281[,1:8,4,2]), LatLong, cex=0.7
```

So we have here 9x4 projections for each PA run, which gives 72 projections per fufture scenario (2 PA runs). So in total : 144 projections.

6.2 Ensemble Forecasting

Several approaches are available for combining ensembles of models in BIOMOD. Here is an example of the use of the *Ensemble.Forecasting* function as well as some details of the different strategies:

Four straightforward means of 'committee averaging' (giving the same weight to all the elements) are done across all the models for each run:

- on the probabilities
- on the binary projection according to the Roc method
- on the binary projection according to the Kappa method
- on the binary projection according to the TSS method

A weighted approach is also available that ranks the models using their evaluation score.
Making a mean on the 0-1 projections gives some sort of probability of occurence. For example, for a given site and with the TSS method, 6 projections give a "1" and 2 give a "0". The mean will be 0.75. It is extracted from binary projection and it is therefore not possible to determine a prior threshold. Conversion into binary is nevertheless possible (see *binary* below).

The median value is also calculated on the probabilities given by the models. It is considered to be more reliable because it is less influenced by extreme values.

Some options:

repetition.models: You can choose to switch on or off the repetition models. If selected, the function will calculate the ensemble forecasts for each run and generate a final one which produces a general ensemble forecast across all the runs for each method. This total consensus is done inconsistently of this argument being set to TRUE or FALSE.

weight.method: the method for ranking the models according to their predictive performance. The *decay* gives the relative importance of the weights. The default weight decay is 1.6; See the example below.

models	GAM	GBM	GLM	ANN	RF	MARS	CTA	FDA
score with Roc	0.96	0.92	0.90	0.88	0.87	0.75	0.72	0.68
decay of 1	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125

decay of 1.2	0.217	0.181	0.151	0.126	0.105	0.087	0.073	0.061
decay of 1.6	0.384	0.240	0.150	0.094	0.059	0.037	0.023	0.014
decay of 2	0.502	0.251	0.125	0.063	0.031	0.016	0.008	0.004

You can type in any value (it has however to be higher than 1) depending on the strength of discrimination that you want. A decay of 1 is equivalent to a committee averaging (i.e. same weights given to all elements).

final.model.out: set to True if you want the total ensemble to be build with the final models taken into account.

qual.th: enables to switch off the models under a certain evaluation score. This will be applied to all models on all species. This option is usefull if you think some of your models are realy bad for your study case.

compress: logical or character string specifying whether saving to a named file is to use compression. FALSE corresponds to no compression, and character strings ”gzip”, or ”xz” specify the type of compression. See ?save for more details. Default is ”xz”. Note that compression may be a long task so you can switch it off if you are more interesting in saving time than in saving space.

R code

```
 Ensemble.Forecasting(Proj.name= "Future1", weight.method='Roc',
                      PCA.median=T, binary=T, bin.method='Roc',
                      Test=F, decay=1.6, repetition.models=T,
                      final.model.out=FALSE, qual.th=0, compress="xz")
```

R code

```
Sp281
Sp290

 consensus_Future1_results
$Sp281
$Sp281$weights
             ANN    CTA    GAM    GBM    GLM   MARS    FDA
PA1       0.0297 0.0143 0.0297 0.1500 0.0762 0.3120 0.0762
PA1_rep1  0.0229 0.0143 0.0586 0.1500 0.0366 0.3839 0.0937
PA1_rep2  0.0297 0.0143 0.0297 0.1219 0.1219 0.2400 0.0586
PA1_rep3  0.0366 0.0143 0.0229 0.1500 0.0937 0.2400 0.0586
PA2       0.0366 0.0229 0.0586 0.2400 0.0143 0.1500 0.0937
PA2_rep1  0.0366 0.0229 0.0586 0.2400 0.0143 0.1500 0.0937
PA2_rep2  0.0366 0.0143 0.0586 0.2400 0.0229 0.1500 0.0937
PA2_rep3  0.0476 0.0229 0.0476 0.0937 0.0143 0.3120 0.1500
              RF SRE
```

```
PA1      0.3120   0
PA1_rep1 0.2400   0
PA1_rep2 0.3839   0
PA1_rep3 0.3839   0
PA2      0.3839   0
PA2_rep1 0.3839   0
PA2_rep2 0.3839   0
PA2_rep3 0.3120   0

$Sp281$PCA.median
         model.selected
PA1      "MARS"
PA1_rep1 "GBM"
PA1_rep2 "GLM"
PA1_rep3 "GLM"
PA2      "MARS"
PA2_rep1 "RF"
PA2_rep2 "GBM"
PA2_rep3 "FDA"

$Sp281$thresholds
                     PA1 PA1_rep1 PA1_rep2 PA1_rep3   PA2
prob.mean          496.8    511.7    451.3    451.6 502.4
prob.mean.weighted 465.1    387.9    370.9    402.7 519.0
median             572.6    609.4    498.2    499.7 576.0
Roc.mean           500.0    500.0    500.0    500.0 500.0
Kappa.mean         500.0    500.0    500.0    500.0 500.0
TSS.mean           500.0    500.0    500.0    500.0 500.0
                   PA2_rep1 PA2_rep2 PA2_rep3
prob.mean             470.4    491.7    454.0
prob.mean.weighted    406.8    421.2    332.1
median                543.2    584.3    487.9
Roc.mean              500.0    500.0    500.0
Kappa.mean            500.0    500.0    500.0
TSS.mean              500.0    500.0    500.0


$Sp290
$Sp290$weights
            ANN    CTA    GAM   GBM    GLM   MARS    FDA
PA1      0.0366 0.0143 0.0937 0.240 0.0586 0.1500 0.0229
PA1_rep1 0.0366 0.0143 0.0937 0.258 0.0586 0.2580 0.0229
PA1_rep2 0.0366 0.0143 0.0937 0.240 0.0586 0.1500 0.0229
PA1_rep3 0.0366 0.0143 0.1950 0.195 0.0762 0.0762 0.0229
             RF SRE
PA1      0.3839   0
PA1_rep1 0.2580   0
PA1_rep2 0.3839   0
PA1_rep3 0.3839   0
```

```
$Sp290$PCA.median
         model.selected
PA1      "GBM"
PA1_rep1 "MARS"
PA1_rep2 "RF"
PA1_rep3 "RF"

$Sp290$thresholds
                      PA1 PA1_rep1 PA1_rep2 PA1_rep3
prob.mean           684.7    651.5    643.5    579.5
prob.mean.weighted 592.3    610.7    602.1    505.2
median              695.1    650.8    568.3    511.8
Roc.mean            500.0    500.0    500.0    500.0
Kappa.mean          500.0    500.0    500.0    500.0
TSS.mean            500.0    500.0    500.0    500.0
```

OUTPUTS

Objects produced : consensus_Future1_results (in Rs memory) which is the list returned by the function. It contains all the computational information that has been used to render the ensemble forecasts, the weights awarded to the models in the weighting process. The model selected by the PCA.median method (if set to True) is also returned and give us the model selected as the first axis of a PCA analyses (that means the model that explain the best the consensus probabilites). The forecasts themselves are stored on the hard disk directly in the corresponding folder.

NOTE1 : For the slot containing the weigths (e.g. *$Sp281$weights*), the PA1 line correspunding to a run calibrate with all the pseudo-absences selected and presences data (models are evaluated on the same data so are often over optimistic). The PA1_rep1, PA1_rep2, and PA1_rep3 lines are linked to models calibrated and validated on two different subset of the pseudo-absences selected and presences data (DataSplit opton in Models).

NOTE2 : The thresholds slot contains some consensus thresholds for differents run. *prob.mean* and *prob.mean.weighted* correspund respectivly to the mean and weighted mean of thresholds used to convert probabilities into presences/abscences data(e.g *(*Evaluation.results.xx) table). *median* is the median of the same thresholds. The values of *Roc.mean, Kappa.mean and TSS.mean* is always set to 500. We made the assumption that as index are resacaled on a 0-1 ladder, 0.5 is the treshold that will discriminate presences and absences.

The function produces an object per species. These objects are arrays of three dimensions :

R code

```
load("proj.Future1/consensus_Sp290_Future1")
dim(consensus_Sp290_Future1)
```

R code

```
[1] 2264    4    6
```

R code

```
dimnames(consensus_Sp290_Future1)[-1]
```

R code

```
[[1]]
[1] "PA1"      "PA1_rep1" "PA1_rep2" "PA1_rep3"

[[2]]
[1] "prob.mean"          "prob.mean.weighted"
[3] "median"             "Roc.mean"
[5] "Kappa.mean"         "TSS.mean"
```

The second dimension is the repetition runs and the third dimension is the consensus methods. There is also an object called ”Total_consensus_Future1” that makes a single output out of all the repetitions.

R code

```
load("proj.Future1/Total_consensus_Future1")
dim(Total_consensus_Future1)
```

R code

```
[1] 2264    2    6
```

R code

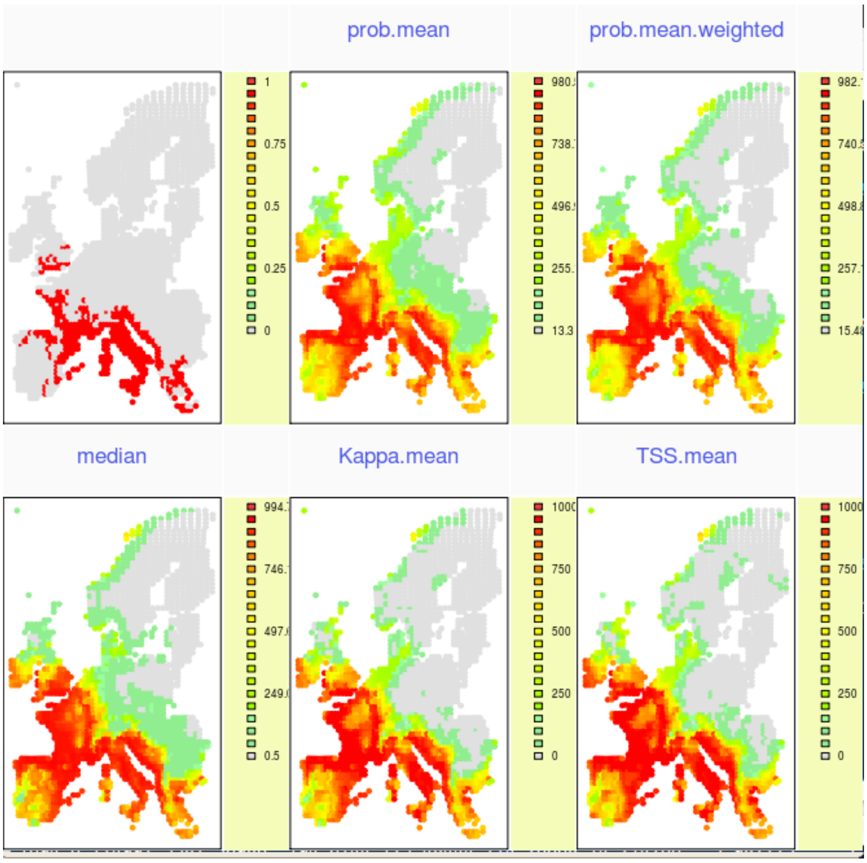
```
dimnames(Total_consensus_Future1)[-1]
```

R code

```
[[1]]
[1] "Sp281" "Sp290"

[[2]]
[1] "prob.mean"          "prob.mean.weighted"
[3] "median"             "Roc.mean"
[5] "Kappa.mean"         "TSS.mean"
```

Now the second dimension is the species. Let's see and plot some of these :

```
R code
Total_consensus_Future1[1:20,,1]
multiple.plot(cbind(DataBIOMOD[,'Sp281'],
                     Total_consensus_Future1[,'Sp281',
                     c(1,2,3,5,6)]), LatLong, cex=0.8)
```



If binary is set to True, the same names are used with a terminal *_Bin* containing the consensus results in binary format.

7 Distributions Changes

7.1 Species Range Change

This function allows to estimate the proportion and relative number of pixels (or habitat) lost, gained and stable for the time slice considered : the range change.

The future range changes are calculated as a percentage of the species' present state. For example, if a species currently occupies 100 cells and is estimated by a model to cover 120 cells in the future, the range change will be + 20%.

The function uses two datasets. The current species distributions and the future one. Note that predictions for current and future must be in a binary (presence and absence) format and in the same resolution.

Let's use our data :

R code

```
load("proj.Future1/Total_consensus_Future1_Bin")
Biomod.RangeSize(CurrentPred = Sp.Env[,c(11,13)],
                 FutureProj = Total_consensus_Future1_Bin[,,2],
                 SpChange.Save="SpChange")
```

A list of two datasets is created: Compt.By.Species and Diff.By.Pixel

Diff.By.Pixel stores useful information for each species. The species are in columns and the pixel in rows. For each species, a pixel could have four different values:
-2 if the given pixel is predicted to be lost by the species.
-1 if the given pixel is predicted to be stable for the species.
0 is the given pixel was not occupied, and will not be into the future.
1 if the given pixel was not occupied, and is predicted to be into the future.

In our examples :

R code

```
SpChange$Diff.By.Pixel[740:760,]
```

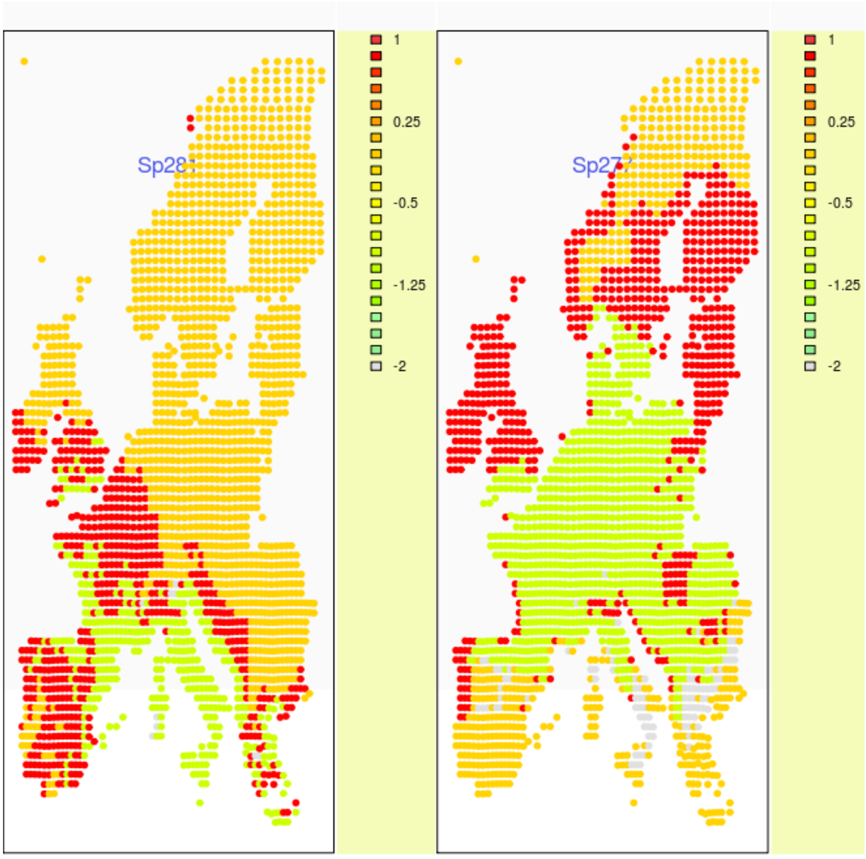
R code

```
    Sp281 Sp277
740    -1    -1
741     1     1
```

```
742    1   -1
743    1    1
744    0   -1
745    1   -1
746    0   -1
747    1   -1
748    1   -1
749    1   -1
750    0   -1
751    0   -1
752   -1    0
753   -1    0
754   -1   -2
755   -1    0
756   -1   -1
757   -1   -1
758   -1   -1
759   -1   -1
760    1    1
```

R code

```
multiple.plot(SpChange$Diff.By.Pixel, LatLong)
```



Compt.By.Species stores the summary of range change for each species (by rows).

The first four columns are relative numbers: Disa represents the number of pixels predicted to be lost by the given species. Stable0 is the number of pixels which are not currently occupied by the given species and not predicted to be. Stable1 represents the number of pixels currently occupied by the given species, and predicted to remain occupied into the future. Gain represent the number of pixels which are currently not occupied by the given species but predicted to be into the future.

PercLoss, PercGain and SpeciesRangeChange are the related percentage estimating as the following:

- CurrentRangeSize represent the modelled current range size (number of pixels occupied) of the given species.
- FutureRangeSize0Disp represents the future modelled range size assuming no migration of the given species.
- FutureRangeSize1Disp represents the future modelled range size assuming migration of the given species (depending on the datasets given in input, if Migration has been used or not).

R code

```
SpChange$Compt.By.Species
```

R code

```
      Loss Stable0 Stable1 Gain PercLoss PercGain
Sp281    9    1326     383  546    2.296    139.3
Sp277   94     550     986  634    8.704     58.7
      SpeciesRangeChange CurrentRangeSize
Sp281                137              392
Sp277                 50             1080
      FutureRangeSize.NoDisp FutureRangeSize.FullDisp
Sp281                    383                      929
Sp277                    986                     1620
```

For other examples, we need some extra species data than the one we have been modelling. Load the dataset called DATA100SP.txt :

R code

```
load("DATA100SP.RData")
#adds three objects : storeC, storeF and Curr
ls()
```

R code

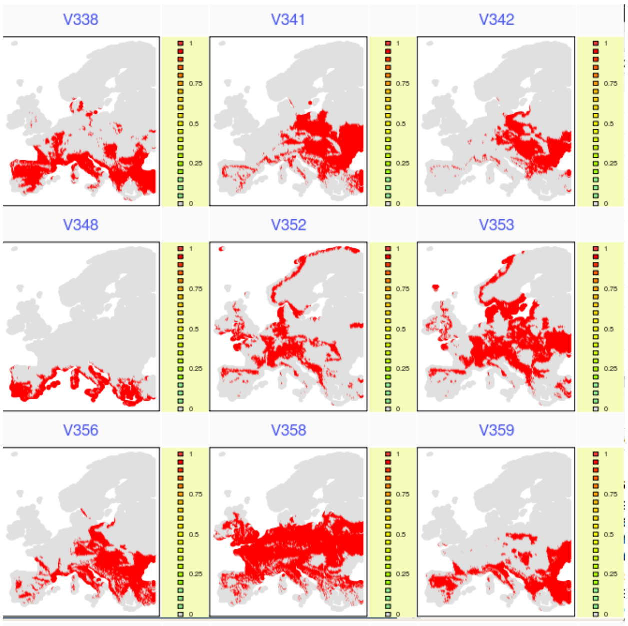
```
 [1] "BestModelByRoc"
 [2] "BestModelByTSS"
 [3] "biomodDependencies"
 [4] "Biomod.material"
 [5] "Biomod.PA.data"
 [6] "Biomod.PA.sample"
 [7] "consensus_Sp290_Future1"
 [8] "Curr"
 [9] "DataBIOMOD"
[10] "DataEvalBIOMOD"
[11] "data.used"
[12] "Evaluation.results.Kappa"
[13] "Evaluation.results.Roc"
[14] "Evaluation.results.TSS"
[15] "Expl.Var"
[16] "Expl.Var2"
[17] "Expl.Var3"
[18] "Future1"
[19] "GBM.list"
[20] "GBM.perf"
[21] "i"
[22] "isnullYweights"
[23] "LatLong"
[24] "missingPackages"
```

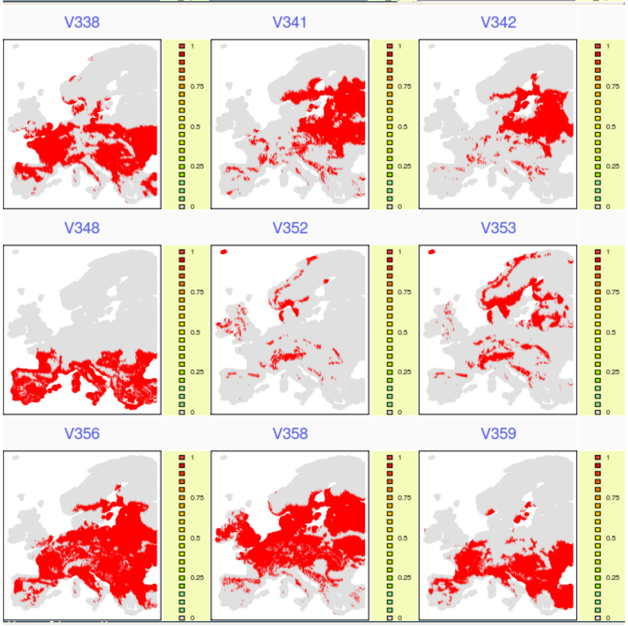
```
[25] "model"
[26] "myPackages"
[27] "obj"
[28] "our.lines"
[29] "Pred"
[30] "Pred2"
[31] "Pred3"
[32] "PredBestModelByKappa"
[33] "Pred_Sp281"
[34] "Pred_Sp290"
[35] "Pred_Sp290_BinKappa"
[36] "Pred_Sp290_FiltKappa"
[37] "Pred_Sp290_indpdt"
[38] "Proj_Future1_Sp290"
[39] "Proj_Future1_Sp290_BinRoc"
[40] "rand"
[41] "Resp.Var"
[42] "Sp290_GLM_PA1"
[43] "Sp290_RF_PA1"
[44] "SpChange"
[45] "Sp.Env"
[46] "store"
[47] "storeC"
[48] "storeF"
[49] "Total_consensus_Future1"
[50] "Total_consensus_Future1_Bin"
[51] "VarImportance"
```

It corresponds to the run of the FDA on 100 species with the same resolution on current and future data, and coordinates (in Curr). Let's have a look at them :

R code

```
multiple.plot(storeC[,1:9], Curr[,1:2], cex=0.7)
multiple.plot(storeF[,1:9], Curr[,1:2], cex=0.7)
```





Let's have a look at the SRC for some of those species :

R code

```
Biomod.RangeSize(CurrentPred = storeC,
                 FutureProj = storeF,
                 SpChange.Save="SpChange100")
SpChange100$Compt.By.Species[1:20,]
```

R code

```
     Loss Stable0 Stable1 Gain PercLoss PercGain
V338 2568   16028    4347 6588    37.14  95.2711
V341 3988   16826    3215 5502    55.37  76.3848
V342 3710   19587     994 5240    78.87 111.3946
V348  306   22084    1840 5301    14.26 247.0177
V352 4426   23197    1269  639    77.72  11.2204
V353 7175   17449    2637 2270    73.12  23.1349
V356  809   13362    6030 9330    11.83 136.4235
V358 3980   11136    9629 4786    29.25  35.1679
V359  951   19655    4556 4369    17.27  79.3354
V360 3641   11938   10053 3899    26.59  28.4723
V365  606    9956   11092 7877     5.18  67.3363
V366 2752   23198    1151 2430    70.51  62.2598
V367 1016   16503    3358 8654    23.23 197.8509
V368 1347   11846    6820 9518    16.49 116.5422
V372 1053   21293    4553 2632    18.78  46.9497
V376  556   15217    4958 8800    10.08 159.5938
V377  591   23803    1602 3535    26.95 161.1947
V379 1817   27081     628    5    74.31   0.2045
V382  396   24532    1750 2853    18.45 132.9450
V385  400   18976    3045 7110    11.61 206.3861
     SpeciesRangeChange CurrentRangeSize
V338             58.134             6915
V341             21.019             7203
V342             32.526             4704
V348            232.759             2146
V352            -66.497             5695
V353            -49.990             9812
V356            124.594             6839
V358              5.923            13609
V359             62.066             5507
V360              1.884            13694
V365             62.156            11698
V366             -8.250             3903
V367            174.623             4374
V368            100.049             8167
V372             28.166             5606
V376            149.510             5514
V377            134.245             2193
V379            -74.110             2445
V382            114.492             2146
V385            194.775             3445
     FutureRangeSize.NoDisp FutureRangeSize.FullDisp
V338                   4347                    10935
V341                   3215                     8717
V342                    994                     6234
V348                   1840                     7141
V352                   1269                     1908
V353                   2637                     4907
```

```
V356                    6030                     15360
V358                    9629                     14415
V359                    4556                      8925
V360                   10053                     13952
V365                   11092                     18969
V366                    1151                      3581
V367                    3358                     12012
V368                    6820                     16338
V372                    4553                      7185
V376                    4958                     13758
V377                    1602                      5137
V379                     628                       633
V382                    1750                      4603
V385                    3045                     10155
```

R code

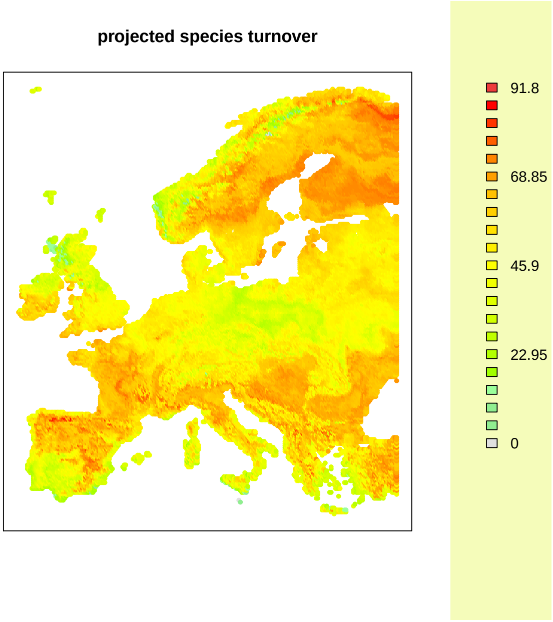
```
samp <- sample(100, 1)
x11()
level.plot(SpChange100$Diff.By.Pixel[,samp], Curr[,1:2], cex=0.6,
                      title=colnames(storeC)[samp])
```

7.2 Species Turnover

This function allows to estimate species loss, gained, and turnover by pixel for the time slice considered.
The function uses two datasets: the current species distributions and a future one. Note that predictions for current and future must be in a binary (presence and absence) format.

We can calculate the projected turnover for the 100 species and produce a plot of the turnover values.

R code
```
Biomod.Turnover(CurrentPred = storeC, FutureProj = storeF,
                     Turnover.Save= "Turnover")
level.plot(Turnover[,7], Curr[,1:2],
                     title='projected species turnover',
                     cex=0.6)
```



R code
```
Turnover[740:750,]
```

R code
```
    Loss Stable0 Stable1 Gain PercLoss PercGain Turnover
740   15      59      18    8    45.45    24.24    56.10
741   15      58      19    8    44.12    23.53    54.76
```

```
742   16      52      21   11    43.24    29.73    56.25
743   10      61      23    6    30.30    18.18    41.03
744    9      61      24    6    27.27    18.18    38.46
745   11      58      25    6    30.56    16.67    40.48
746    9      59      26    6    25.71    17.14    36.59
747    9      59      27    5    25.00    13.89    34.15
748   10      57      26    7    27.78    19.44    39.53
749    8      60      26    6    23.53    17.65    35.00
750   10      58      25    7    28.57    20.00    40.48
    CurrentSR FutureSR.NoDisp FutureSR.FullDisp
740        33              18                26
741        34              19                27
742        37              21                32
743        33              23                29
744        33              24                30
745        36              25                31
746        35              26                32
747        36              27                32
748        36              26                33
749        34              26                32
750        35              25                32
```

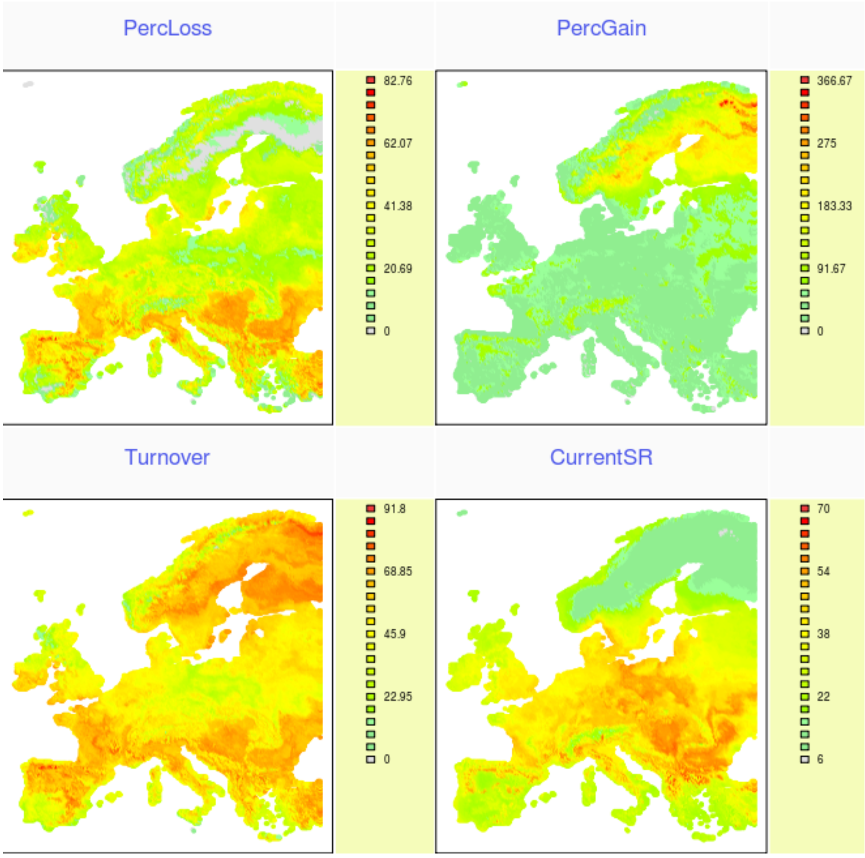
In the stored database, 10 columns are created.
The first four columns are relative numbers: Disa represents the number of species predicted to disappear from the given pixel. Stable0 is the number of species which are currently not in the given pixel and not predicted to migrate. Stable1 represents the number of species currently occurring in the given pixel, and predicted to remains into the future. Gain represent the number of species which are currently absent but predicted to migrate in the given pixel.

PercLoss, PercGain and Turnover are the related percentage estimated as the following:
- PercLoss = 100 x L/(SR)
- PercGain = 100 x G/(SR)
- Turnover = 100 x (L+G)/(SR+G)
Where SR is the current species richness.

CurrentSR represent the current modelled species richness in the given pixel.
FutureSR0Disp represents the future modelled species richness assuming no migration of species
FutureSR1Disp represents the future modelled species richness assuming migration (depending on the datasets given in input, if Migration has been used or not).

R code

```
multiple.plot(Turnover[,5:8], Curr[,1:2], cex=0.6)
```



7.3 Probability Density Function

This function enables an overall viewing of the future projections range per species and gives the likelihood of range shift estimations. The optimal way for condensing 50, 75, 90 and 95% of the data will be calculated.

initial: a vector in a binary format (ones and zeros) representing the current distribution of a species which will be used as a reference for the range change calculations.

projection: a matrix grouping all the projections where each column is a single prediction. Make sure you keep projections in the same order as the initial vector (line1=site1, line2=site2, etc.).

Resolution: the step used for classes of projection in graphics. The default value is 5.

NOTE: modifying the resolution will directly influence the probability scale. Bigger classes will cumulate a greater number of predictions and therefore represent a greater fraction of the total predictions. The probability is in fact that of the class and not of isolated events.

cvsn: stands for current vs new. If true, the range change calculations will be of two types: the percentage of cells currently occupied by the species to be lost, and the relative percentage of cells currently unoccupied but projected to be, namely 'new' cells, compared to current surface range.

With the example above where the species will have 120 suitable sites in the future whilst only 100 at present, this might be the result of different events. A case could be that the 100 present cells are kept and an additional 20 new sites makes the 120 cells. Another possibility is that the 100 current cells are predicted to be lost with 120 new cells, also giving 120 total cells in future.

These two cases bring the same SRC calculations results, but whilst the first case does not imply much as in survival strategies (the current populations will still be in good conditions in future, plus even having new potential territories to explore and colonise), the second case, however, implies a strong migrating effort for the populations to stay in suitable environments. Those two cases and all in-between possibilities are distinguishable with this method.

groups: an option for ungrouping the projections enabling a separated visualisation of the prediction range per given group. A matrix is expected where each column is a single projection and each line is giving details of one parameter.

Do keep in mind that this matrix represents the projections the way you have put them into the *projection* argument. Sort your matrix the way you have sorted your projections!

In can look like this:

R code

```
     [,1]  [,2]    [,3]  [,4]  [,5]    [,6]  [,7]  [,8]
[1,] "GAM" "GAM"   "GAM" "CTA" "CTA"   "CTA" "RF"  "RF"
[2,] "Roc" "Kappa" "TSS" "Roc" "Kappa" "TSS" "Roc" "Kappa"
     [,9]
[1,] "RF"
[2,] "TSS"
```

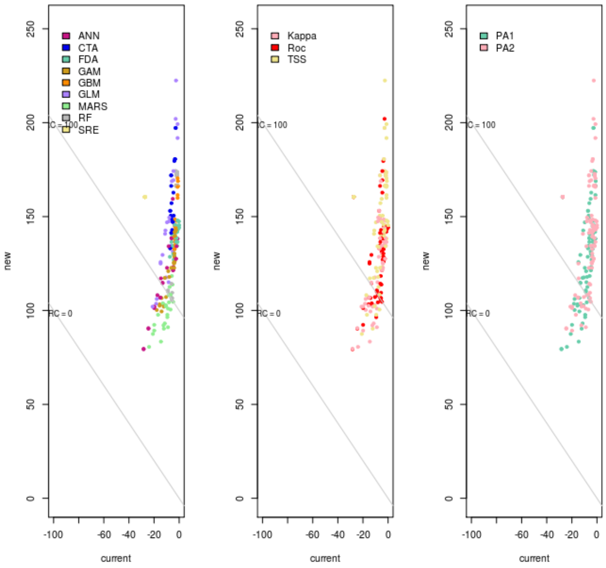
R code

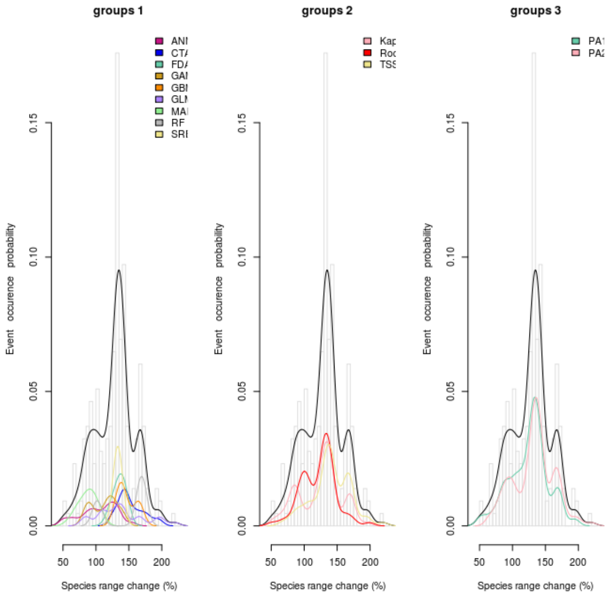
```
 #preparation of data
 ## Sp281
 scenarios <- "Future1"
 models <- c("ANN","CTA","GAM","GBM","GLM","MARS","FDA","RF","SRE")
 evaluation <- c("Roc", "Kappa", "TSS")
 reps <- 4
 PAs <- c("PA1", "PA2")
 DataFrame <- matrix(NA, 2264, 2)
 Groups <- matrix(NA, 3, 216)
 Groups[1,] <- c(rep(models,24))
 Groups[2,] <- c(rep(rep(evaluation, each=36),2 ))
 Groups[3,] <- c(rep(rep(PAs, each=108), 1))
 for(sc in scenarios){
      for(PA in PAs){
          for(ev in evaluation){

              eval(parse(text=paste("load('proj.", sc, "/Proj_", sc,
                                    "_Sp281_Bin", ev, "')", sep="")))
              add.data <- eval(parse(text=paste("Proj_", sc, "_Sp281_Bin",
                                                ev, sep="")))

              DataFrame <- cbind(DataFrame, add.data[,, 'total.data', PA])
              DataFrame <- cbind(DataFrame, add.data[,, 'rep1', PA])
              DataFrame <- cbind(DataFrame, add.data[,, 'rep2', PA])
              DataFrame <- cbind(DataFrame, add.data[,, 'rep3', PA])
          }
      }
 }
```

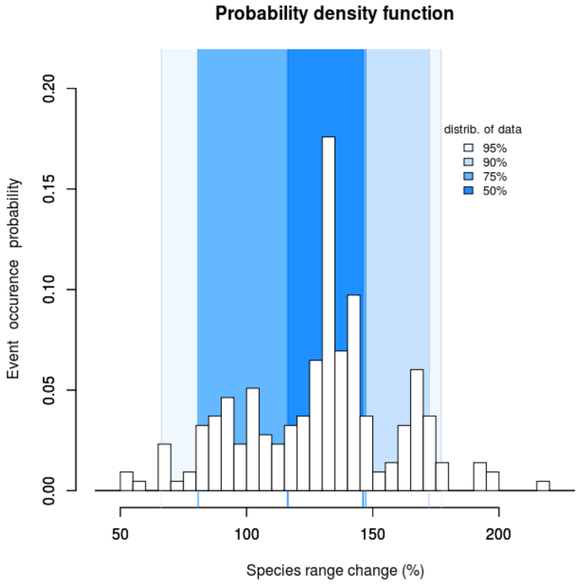
```
PDFdata281 <- DataFrame[,-1][,-1]
ProbDensFunc(initial=Sp.Env[,'Sp281'], projections=PDFdata281, cvsn = T,
              groups = Groups, resolution = 5)
```

R code

```
$stats
    lower limit upper limit
50%      116.33       146.2
75%       80.87       147.2
90%       80.87       172.2
95%       66.07       177.3
```







The two lines represent where the SRC value is 0 (no absolute change in the number of suitable sites) and +100% (the species will double its current potential distribution size). Along those line, you have all the possibilities for giving that one value (-10+10=0 ; -40+40=0 ; ...).

On the cvsn graph, each dot is a projection. See how the single SRC value does not reflect every thing that is going on. In certain cases it hides the potential loss of current habitats, which would surely lead to different management decisions if known.

7.3.1 An example with repetitions

The help file of the ProbDensFunc function provides a full example. It is done with 20 repetitions for half of the models to assess the variability in prediction making when the calibration of the model is done on partial data. Only Sp163 is done. Please look in details the help file for an example of the data preparation you should go through to run the function properly.

R code

```
example(ProbDensFunc)
```