Multi-species distribution modeling with `biomod2`

`biomod2` version : 2.1.0
R version 2.15.2 (2012-10-26)

Damien Georges & Wilfried Thuiller

February 22, 2013

Contents

1   Introduction

This vignette illustrates how to model several species all together with `biomod2`.
In `biomod2` almost all the functions are built to work on single species. This choice have been made to facilitate parallel computing on large dataframes and rasters.
The aim of this document is to show how to perform multi-species modeling with `biomod2`. This has many advantages and is not a difficult task.

In the next sections, we will provide examples on how to model a set of species and an introduction to parallel programming.

NOTE 1 :
We plan to develop in the near future additional functions to help users to deal with this multi-species modeling approach.

2   Multiple Species Computation

Functions allowing to model multiple species are not implemented yet. Thus, you will need to create
a 'loop' around the functions related to the modeling steps in `biomod2` (e.g, initialisation, modeling,
ensemble modeling, projection, ensemble forecasting) . . . The best way for doing this task depends on
the class of your input data. Let's start with with the `biomod2` data.

2.1   Loading the data

First, we need to load our species occurrence data. Here, species occurrences have been previously
extracted and stored as a table (.csv format).

```
─────────────────────────────── R input ───────────────────────────────
# 1. loading species occurrences data
# 1. loading species occurrences data
library(biomod2)
DataSpecies <- read.csv( system.file(
                         "external/species/mammals_table.csv",
                         package="biomod2"))
head(DataSpecies)
```

```
─────────────────────────────── R output ───────────────────────────────
  X X_WGS84 Y_WGS84 ConnochaetesGnou GuloGulo PantheraOnca
1 1   -94.5      82                0        0            0
2 2   -91.5      82                0        1            0
3 3   -88.5      82                0        1            0
4 4   -85.5      82                0        1            0
5 5   -82.5      82                0        1            0
6 6   -79.5      82                0        1            0
  PteropusGiganteus TenrecEcaudatus VulpesVulpes
1                 0               0            0
2                 0               0            0
3                 0               0            0
4                 0               0            0
5                 0               0            0
6                 0               0            0
```

```
─────────────────────────────── R input ───────────────────────────────
```

In this example, we have presences/absences data and correspunding coordinates for 6 mammals.
Our explanatory variables are stored in a RasterStack.

```
─────────────────────────────── R input ───────────────────────────────
# 2. loading environmental data

# Environmental variables extracted from Worldclim (bio_3, bio_4,
# bio_7, bio_11 & bio_12)
require(raster)
myExpl = stack( system.file( "external/bioclim/current/bio3.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio4.grd",
                             package="biomod2"),
                system.file( "external/bioclim/current/bio7.grd",
                             package="biomod2"),
```

```
                   system.file( "external/bioclim/current/bio11.grd",
                                package="biomod2"),
                   system.file( "external/bioclim/current/bio12.grd",
                                package="biomod2"))
```

We model the niches of 6 mammal species having quite different distributions:

- ConnochaetesGnou

- GuloGulo

- PantheraOnca

- PteropusGiganteus

- TenrecEcaudatus

- VulpesVulpes

2.2   Running the models with the for function

For each species, we will sequentially follow these steps:

1. Selecting data corresponding to a species

2. Putting this data in the `biomod2` format (i.e. BIOMOD_FormatingData)

3. Building 'individual models' (i.e. BIOMOD_Modeling)

4. Building ensemble-models (i.e. BIOMOD_EnsembleModeling)

5. Making model projections (i.e. BIOMOD_Projection BIOMOD_EnsembleForecasting)

NOTE 2 :
The modeling steps applied to each species are the same as the one described in 'GettingStarted' vignette.

```
_____ R input _____
# define the species of interest
sp.names <- c("ConnochaetesGnou", "GuloGulo", "PantheraOnca", "PteropusGiganteus",
              "TenrecEcaudatus", "VulpesVulpes")
# loop on species == applying the same functions to each species
for(sp.n in sp.names){
  myRespName = sp.n

  cat('\n',myRespName,'modeling...')
  ### definition of data
  ## i.e keep only the column of our species
  myResp <- as.numeric(DataSpecies[,myRespName])

  myRespCoord = DataSpecies[c('X_WGS84','Y_WGS84')]



  ### Initialisation
  myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                       expl.var = myExpl,
```

```
                                    resp.xy = myRespCoord,
                                    resp.name = myRespName)


### Options definition
myBiomodOption <- BIOMOD_ModelingOptions()

### Modelling
myBiomodModelOut <- BIOMOD_Modeling(
                    myBiomodData,
                    models = c('SRE','CTA','RF','MARS','FDA'),
                    models.options = myBiomodOption,
                    NbRunEval=3,
                    DataSplit=80,
                    Prevalence=0.5,
                    VarImport=3,
                    models.eval.meth = c('TSS','ROC'),
                    SaveObj = TRUE,
                    rescal.all.models = TRUE,
                    do.full.models = FALSE,
                    modeling.id = paste(myRespName,"FirstModeling",sep=""))

### save models evaluation scores and variables importance on hard drive
capture.output(getModelsEvaluations(myBiomodModelOut),
    file=file.path(myRespName,
                paste(myRespName,"_formal_models_evaluation.txt", sep="")))

capture.output(getModelsVarImport(myBiomodModelOut),
    file=file.path(myRespName,
                paste(myRespName,"_formal_models_variables_importance.txt", sep="")))


### Building ensemble-models
myBiomodEM <- BIOMOD_EnsembleModeling(
                modeling.output = myBiomodModelOut,
                chosen.models = 'all',
                em.by='all',
                eval.metric = c('TSS'),
                eval.metric.quality.threshold = c(0.7),
                prob.mean = T,
                prob.cv = T,
                prob.ci = T,
                prob.ci.alpha = 0.05,
                prob.median = T,
                committee.averaging = T,
                prob.mean.weight = T,
                prob.mean.weight.decay = 'proportional' )

### Make projections on current variable
myBiomomodProj <- BIOMOD_Projection(
                    modeling.output = myBiomodModelOut,
                    new.env = myExpl,
                    proj.name = 'current',
                    selected.models = 'all',
                    binary.meth = 'TSS',
                    compress = 'xz',
                    clamping.mask = F,
```

```
                        output.format = '.grd')

   ### Make ensemble-models projections on current variable
   myBiomodEF <- BIOMOD_EnsembleForecasting(
                   projection.output = myBiomomodProj,
                   EM.output = myBiomodEM,
                   binary.meth = 'TSS')
}
```

biomod2 creates a folder for each modelled species. You can find all outputs in your working directory. You are now able to work on the outputs and combine them as you want.

To illustrate it let's create an α-diversity map (in this example, this is simply the sum of the binary outputs). We chose to consider only the binary files of 'total consensus ensemble-models projections' and focus our attention on the 'mean of probability' ensemble-model.

```
─────────────────────────────── R input ───────────────────────────────
# define a mask of studied
alphaMap <- reclassify(subset(myExpl,1), c(-Inf,Inf,0))
# # add all other species map
for(sp.n in sp.names){
   # add layer
   alphaMap <-
     alphaMap +
     subset(stack(file.path(sp.n,
                         "proj_current",
                         paste("proj_current_",
                             sp.n,
                             "_TotalConsensus_EMbyTSS_TSSbin.grd", sep=""))), 1)
 }
# summary of created raster
alphaMap
```

```
─────────────────────────────── R output ───────────────────────────────
class       : RasterLayer
dimensions  : 47, 120, 5640   (nrow, ncol, ncell)
resolution  : 3, 3  (x, y)
extent      : -180, 180, -57.5, 83.5   (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : in memory
names       : layer
values      : 0, 3   (min, max)
```
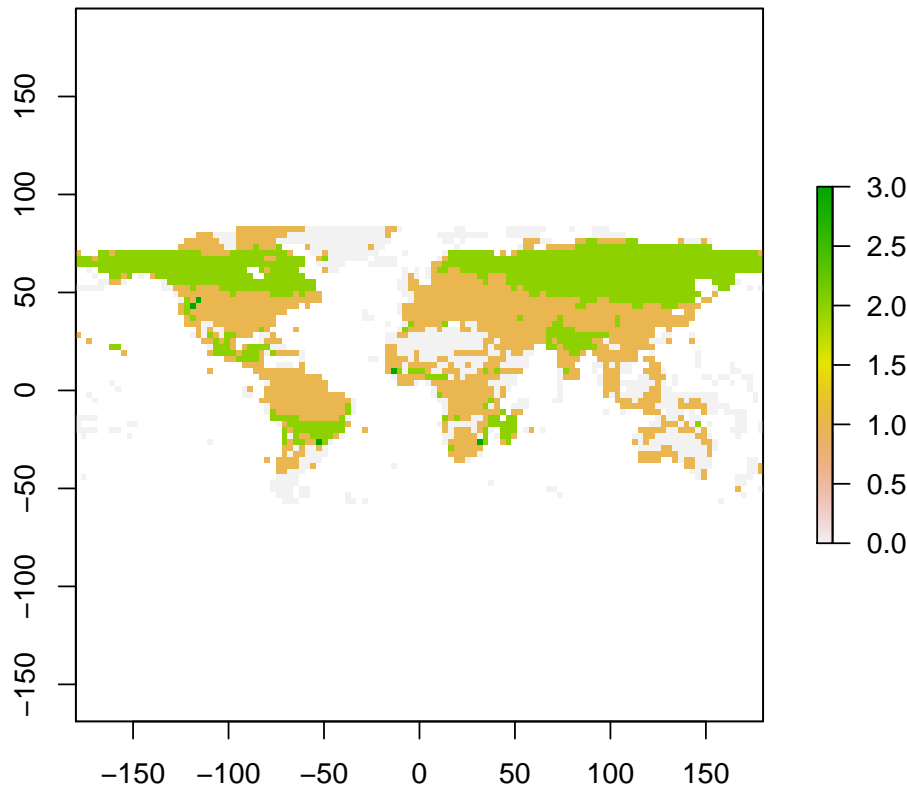
```
─────────────────────────────── R input ───────────────────────────────
```

Let's visualise our α-diversity map.

```
─────────────────────────────── R input ───────────────────────────────
plot(alphaMap, main = expression( paste(alpha, "-diversity based on",
                       " TotalConsensus_EMbyTSS_TSSbin outputs")))
```

α−diversity based on TotalConsensus_EMbyTSS_TSSbin outputs

NOTE 3 :
This example is presented with a loop. A more efficient way of programming is to use the lapply function, an inner R function for looping over objects.

2.3   Running the models with the lapply function

lapply is a very easy function to use. It only requires a vector with the species names and a function to run. Here, we just integrate the loop into MyBiomodSF function.

```
───────────────────────────── R input ─────────────────────────────
MyBiomodSF <- function(sp.n){

    myRespName = sp.n

    cat('\n',myRespName,'modeling...')
    ### definition of data
    ## i.e keep only the column of our species
    myResp <- as.numeric(DataSpecies[,myRespName])

    myRespCoord = DataSpecies[c('X_WGS84','Y_WGS84')]


    ### Initialisation
```

```
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespCoord,
                                     resp.name = myRespName)



### Options definition
myBiomodOption <- BIOMOD_ModelingOptions()

### Modelling
myBiomodModelOut <- BIOMOD_Modeling(
  myBiomodData,
  models = c('SRE','CTA','RF','MARS','FDA'),
  models.options = myBiomodOption,
  NbRunEval=3,
  DataSplit=80,
  Prevalence=0.5,
  VarImport=3,
  models.eval.meth = c('TSS','ROC'),
  SaveObj = TRUE,
  rescal.all.models = TRUE,
  do.full.models = FALSE,
  modeling.id = paste(myRespName,"FirstModeling",sep=""))

### save models evaluation scores and variables importance on hard drive
capture.output(getModelsEvaluations(myBiomodModelOut),
    file=file.path(myRespName,
                   paste(myRespName,"_formal_models_evaluation.txt", sep="")))

capture.output(getModelsVarImport(myBiomodModelOut),
    file=file.path(myRespName,
                   paste(myRespName,"_formal_models_variables_importance.txt", sep="")))


### Building ensemble-models
myBiomodEM <- BIOMOD_EnsembleModeling(
  modeling.output = myBiomodModelOut,
  chosen.models = 'all',
  em.by='all',
  eval.metric = c('TSS'),
  eval.metric.quality.threshold = c(0.7),
  prob.mean = T,
  prob.cv = T,
  prob.ci = T,
  prob.ci.alpha = 0.05,
  prob.median = T,
  committee.averaging = T,
  prob.mean.weight = T,
  prob.mean.weight.decay = 'proportional' )

### Make projections on current variable
myBiomomodProj <- BIOMOD_Projection(
  modeling.output = myBiomodModelOut,
  new.env = myExpl,
  proj.name = 'current',
  selected.models = 'all',
  binary.meth = 'TSS',
```

```
    compress = 'xz',
    clamping.mask = F,
    output.format = '.grd')

  ### Make ensemble-models projections on current variable
  myBiomodEF <- BIOMOD_EnsembleForecasting(
    projection.output = myBiomomodProj,
    EM.output = myBiomodEM,
    binary.meth = 'TSS')

}
```

Then, simply call the lapply function and give the vector of species names

```
────────────────────────────────── R input ──────────────────────────────────
myLapply_SFModelsOut <- lapply( sp.names, MyBiomodSF)
```

3   Parallel computing

If you have a large range of species or/and if your data are extremely large , computation time may rapidly expand and become too long to be done within a loop.
The best way to deal with such cases is to do parallel programming. It means modeling species in different CPUs. At the end you will need to re-assemble together the outputs created especially if you aim at doing 'community' analyses such as calculating turnover or species richness.
In the best case you have an access to a cluster of computers. If not, you can have several computers and split your species into groups by hand. But you can also just have a single computer with several CPUs. In such case, you can easily parallelize on your multi-CPU single computer. The snowfall package will help you doing this.
NOTE 4 :
Parallel computing tools are developing fast in R, so whatever the type of parallel programming you want to do, tools may exist to help you! (look at packages snow, snowfall, xgrid, parallel)

4   Introduction to snowfall library

The snowfall package is useful to perform parallel computing with R jobs. The syntax is quite similar to the "lapply()" function we just presented above. The first thing to do is to download the snowfall library.

```
────────────────────────────────── R input ──────────────────────────────────
install.packages('snowfall', dependencies=TRUE)
```

```
────────────────────────────────── R input ──────────────────────────────────
library(snowfall)
```

The second thing to do is to create the function to parallelize (let's take the same as in the previous section). Here, we used the MyBiomodSF function created in the lapply presentation.
Then you have to give to snowfall all the variables and libraries that are used in the function.

```R
─────────────────────────────────── R input ───────────────────────────────────
## Init snowfall
library(snowfall)
sfInit(parallel=TRUE, cpus=2 )  ## we select 2 CPUs. If you have 8 CPUs, put 8.
## Export packages
sfLibrary('biomod2', character.only=TRUE)
## Export variables
sfExport('mySpeciesOcc')
sfExport('myExpl')
sfExport('sp.names')
# you may also use sfExportAll() to export all your workspace variables

## Do the run
mySFModelsOut <- sfLapply( sp.names, MyBiomodSF)
## stop snowfall
sfStop( nostop=FALSE )
```

You will obtain exactly the same folder structure than if you worked in serial so you can compute your 'community' analyses exactly as previously.
NOTE 5 :
In case of computation on several computers, you have to merge all your species folders in a single folder, then you will get the same files as in a serial computation case.