Multi-species distribution modeling with `biomod2`

`biomod2` version : 1.0
R version 2.15.1 (2012-06-22)

Damien Georges & Wilfried Thuiller

July 5, 2012

Contents

1 Introduction **3**

2 Multiple Species Computation **4**
2.1 Loading the data 4

3 How to saving computation time? **8**

4 Introduction to snowfall library **9**

1 Introduction

This vignette illistrates how to model multi-species with `biomod2`.
In `biomod2` almost all the functions are built to work on monospecific species.
This choice have been made to facilitate parallel computing.
The aim of this dcument 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 an example of how to model multi-species and an introduction for parallel programming.

NOTE 1 :
We plan to develop in the near future some tools 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 that will depend on the class of your input data. Let's have an example with `biomod2` data.

2.1 Loading the data

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

R input

```
# 1. loading species occurrences data
library(biomod2)
mySpeciesOcc <- read.csv( system.file(
                            "external/species/species_occ.csv",
                            package="biomod2"))
head(mySpeciesOcc)
```

R output

```
       x     y CapraIbex MelesMeles MyocastorCoypus
1 21.667 68.33       NA          1             NA
2 25.000 68.33       NA          1             NA
3 28.333 68.33       NA          1             NA
4 31.667 68.33       NA          1             NA
5  8.333 65.00       NA          1             NA
6 11.667 65.00       NA          1             NA
```

NOTE 2 :
The loaded table contains only 1 and NA's. Later, we will consider only the occurrences (i.e. 1) for selecting pseudo-absences into our explanatory variables (RasterStack). You can find more details about input data supported by `biomod2` in the 'SupportedInputData' vignette.

In this example, our explanatory variables are stored in a RasterStack.

R input

```
# 2. loading environemental data

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

```
                    system.file( "external/climat/current/bio7.grd",
                                 package="biomod2"),
                    system.file( "external/climat/current/bio11.grd",
                                 package="biomod2"),
                    system.file( "external/climat/current/bio12.grd",
                                 package="biomod2"))
```

We will model the niches of 2 species characterized by their names :

- MelesMeles
- MyocastorCoypus

For each species, we will sequentially follow these steps:

1. Selecting data corresponding to a species
2. Putting this data in the `biomod2` format and selecting pseudo-absences (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 3 :
The modeling steps applied to each species are the same as the one described in 'GettingStarted' vignette.

R input

```
# define the species you want
sp.names <- c("MelesMeles", "MyocastorCoypus")
# loop on species == aplying the same functions to each species
for(sp.n in sp.names){

  cat('\n',sp.n,'modeling...')
  ### definition of data for this run
  ## i.e keep only the column of our species
  myResp <- as.numeric(mySpeciesOcc[,sp.n])
  # get NAs id
  na.id <- which(is.na(myResp))
  # remove NAs to enforce PA sampling to be done on explanatory rasters
  myResp <- myResp[-na.id]

  myRespCoord = mySpeciesOcc[-na.id,c('x','y')]

  myRespName = sp.n
```

```
### Initialisation
myBiomodData <- BIOMOD_FormatingData(resp.var = myResp,
                                     expl.var = myExpl,
                                     resp.xy = myRespCoord,
                                     resp.name = myRespName,
                                     PA.nb.rep = 2,
                                     PA.nb.absences = 10*sum(myResp==1,
                                                             na.rm=TRUE),
                                     PA.strategy = 'random')


### Options definition
myBiomodOption <- BIOMOD_ModelingOptions()

### Modelling
myBiomodModelOut <- BIOMOD_Modeling(
                            myBiomodData,
                            models = c('SRE','CTA','RF','MARS','FDA'),
                            models.options = myBiomodOption,
                            NbRunEval=1,
                            DataSplit=80,
                            Yweights=NULL,
                            VarImport=3,
                            models.eval.meth = c('TSS','ROC'),
                            SaveObj = TRUE,
                            rescal.all.models = TRUE)

### Building ensemble-models
myBiomodEM <- BIOMOD_EnsembleModeling(
                        modeling.output = myBiomodModelOut,
                        chosen.models = 'all',
                        eval.metric = c('TSS'),
                        eval.metric.quality.threshold = c(0.85),
                        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' )

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

```
                                compress = 'xz',
                                clamping.mask = F)

    ### Do ensemble-models projections on current varaiable
    myBiomodEF <- BIOMOD_EnsembleForecasting(
                            projection.output = myBiomomodProj,
                            EM.output = myBiomodEM,
                            binary.meth = 'TSS',
                            total.consensus = TRUE)
  }
```

`biomod2` creates a folder for each species modeled that you can find in following 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 (which in this example will be a sum of the binary outputs). We chose to consider only the binary files of 'total consensus ensemble-models projections' and focus our attention on 'mean of probability' ensemble-models.

R input

```
# load the first speces binary maps which will define the mask
alphaMap <- get(load(paste(sp.names[1],"/proj_current/",
                           sp.names[1],"_TotalConsensus.bin.TSS",
                           sep="")))[[1]]
# free space
rm(list=paste(sp.names[1],"_TotalConsensus.bin.TSS", sep=""))
# # add all other species map
for(sp.n in sp.names[-1]){
   # add layer
   alphaMap <- alphaMap + get(load(paste(sp.n,"/proj_current/",
                                         sp.n,"_TotalConsensus.bin.TSS",
                                         sep="")))[[1]]
   # free space
   rm(list=paste(sp.n,"_TotalConsensus.bin.TSS", sep=""))
 }
# summary of created raster
alphaMap
```

R output

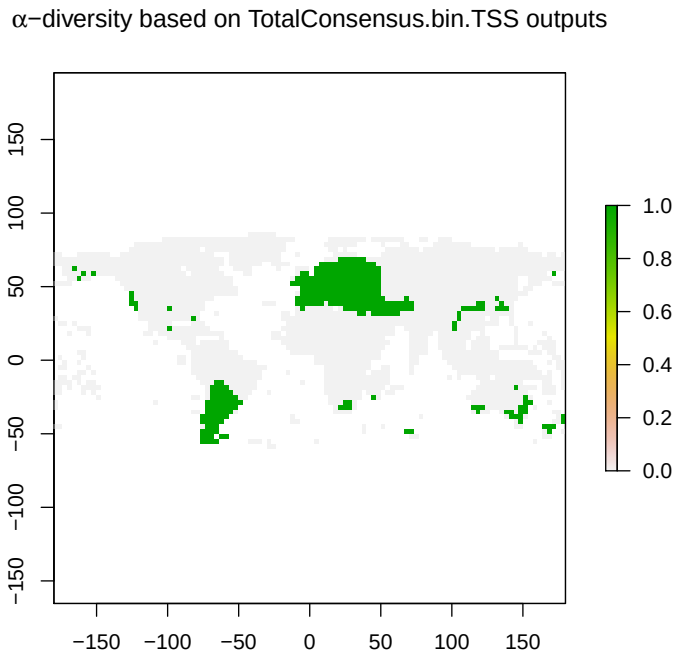
```
class       : RasterLayer
dimensions  : 45, 108, 4860  (nrow, ncol, ncell)
resolution  : 3.333, 3.333  (x, y)
extent      : -180, 180, -60, 90  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0
values      : in memory
min value   : 0
max value   : 1
```

R input

Let's visualise our α-diversity map.

R input

```
plot(alphaMap, main = expression( paste(alpha, "-diversity based on",
                                        " TotalConsensus.bin.TSS outputs")))
```



NOTE 4 :
Here, our two species seem to never coexist, this explains why the α-diversity is never higher than 1.

3 How to saving computation time?

If you have lots of species to study or/and if you work with huge dataset, 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 spliting species in several small groups and model them independently. 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 5 :
Parallel computing tools are developping fast in R so whatever the type of parallel programming you want to do, tools may exist to help you!

4 Introduction to snowfall library

The snowfall package is usefull to perform parallel computing with R jobs. The syntax is quite similar to the "apply()" function. The first thing to do is to download snowfall library.

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

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

The second thing to do is to create the function you will parallelize (let's take the same as in the previous section). Here, we just integrate the loop into MyBiomodSF function.

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

    cat('\n',sp.n,'modeling...')
    ### definition of data for this run
    ## i.e keep only the column of our species
    myResp <- as.numeric(mySpeciesOcc[,sp.n])
    # get NAs id
    na.id <- which(is.na(myResp))
    # remove NAs to enforce PA sampling to be done on explanatory rasters
    myResp <- myResp[-na.id]

    myRespCoord = mySpeciesOcc[-na.id,c('x','y')]

    myRespName = sp.n

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

```
                                        expl.var = myExpl,
                                        resp.xy = myRespCoord,
                                        resp.name = myRespName,
                                        PA.nb.rep = 2,
                                        PA.nb.absences = 10*sum(myResp==1,
                                                                na.rm=TRUE),
                                        PA.strategy = 'random')


### Options definition
myBiomodOption <- BIOMOD_ModelingOptions()

### Modelling
myBiomodModelOut <- BIOMOD_Modeling(
                            myBiomodData,
                            models = c('SRE','CTA','RF','MARS','FDA'),
                            models.options = myBiomodOption,
                            NbRunEval=1,
                            DataSplit=80,
                            Yweights=NULL,
                            VarImport=3,
                            models.eval.meth = c('TSS','ROC'),
                            SaveObj = TRUE,
                            rescal.all.models = TRUE)

### Building ensemble-models
myBiomodEM <- BIOMOD_EnsembleModeling(
                        modeling.output = myBiomodModelOut,
                        chosen.models = 'all',
                        eval.metric = c('TSS'),
                        eval.metric.quality.threshold = c(0.85),
                        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' )

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

```
    ### Do ensemble-models projections on current varaiable
    myBiomodEF <- BIOMOD_EnsembleForecasting(
                          projection.output = myBiomomodProj,
                          EM.output = myBiomodEM,
                          binary.meth = 'TSS',
                          total.consensus = TRUE)

  }
```

Then you have to give to snowfall all the variables and libraries that are used in the function.

R input

```
## Init snowfall
library(snowfall)
sfInit(parallel=TRUE, cpus=2 )
## 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 exacly the same folder structure than if you worked in serial so you can compute your 'community' analyses exactly as previously.
NOTE 6 :
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.