getting started with sdm package

Babak Naimi, Miguel B. Araujo

3 April 2016

sdm is an object-oriented, reproducible and extensible R platform for species distribution modelling. The sdm package is designed to create a comprehensive modelling and simulation framework that: 1) provides a standardised and unified structure for handling species distributions data and modelling techniques (e.g. a unified interface is used to fit different models offered by different packages); 2) is able to support markedly different modelling approaches; 3) enables scientists to modify the existing methods, extend the framework by developing new methods or procedures, and share them to be reproduced by the other scientists; 4) handles spatial as well as temporal data for single or multiple species; 5) employs high performance computing solutions to speed up modelling and simulations, and finally; 6) uses flexible and easy-to-use GUI interface. For more information, check the published paper by Naimi and Araujo (2016) in the journal of Ecography.

This document provides a very quick demonstration on the package followed by some examples, that would be helpful to get a guick start with the package.

Installing sdm and all the required packages

sdm can simply be installed using the standard install.packages function as:

install.packages('sdm')

Depending on the methods are selected through the modelling and using the package, several packages may be needed, and therefore, should be installed on your machine. A quick way to install all the required packages (to guarantee having full functionaliy of sdm), is to simply use the function **installAll** offered by the sdm package. You can simply call it without any argument:

installAll()

A brief overview:

There are three main functions provide the main steps of developing/using species distibution models. The three steps include data preparation, model fitting/ evaluation, and prediction. The functions used for these steps:

- **sdmData**: to read data from different formats, and prepare a data object. Both species (single or multiple) and explanatory variables can be introduced to the function, as well as other information such as spatial coordinates, time, grouping variables, etc.
- **sdm**: to fit and evaluate species distribution models (multiple algorithms can be used)
- **predict**: when models are fitted, they can be used to predict/project given a new dataset.

Example Dataset:

The package is followed by several datasets that are used in the help pages. We also use one of those examples here for our demonstration:

We use a shapefile containing presence=absence records for a species as spatial points (species.shp), and four raster datasets (in Ascii format) as explanatory variables (predictors). The files are in the sdm library, so

we can directly read them from the library folder. We use rgdal to read raster data (it supports different common formats), and raster package to handle the raster data.

```
library(sdm)

library(raster)
library(rgdal)

file <- system.file("external/species.shp", package="sdm") # get the location of the species shapefile

# so, file is simply a filename (with path):
file
```

```
## [1] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/sdm/external/species.shp"
```

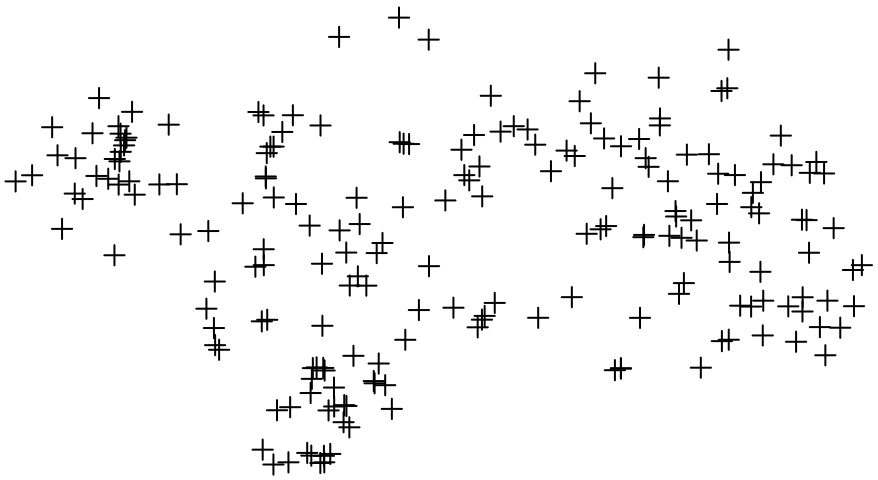
```
# read the species shapefile using the function shapefile:

species <- shapefile(file)

class(species) # it is a SpatialPointsDataFrame
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
plot(species)
```



```
# we can take a look at the head of attribute table in the species dataset:

head(species)
```

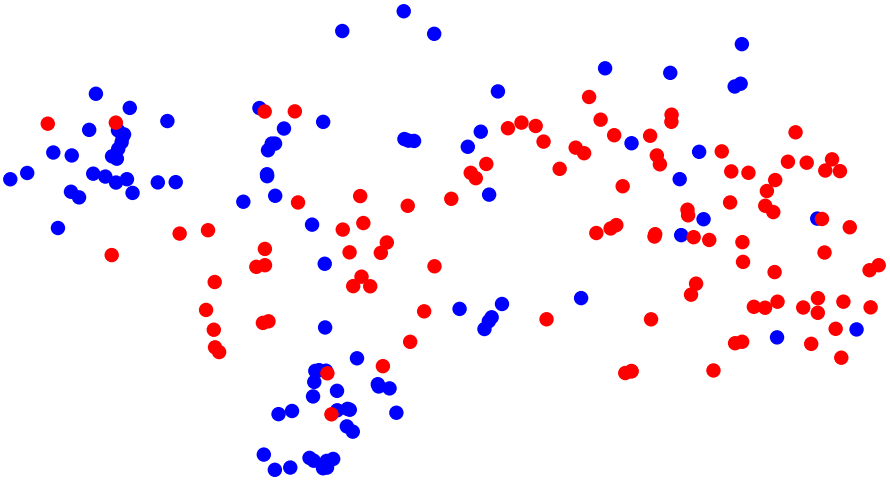
```
##   Occurrence
## 1          1
## 2          0
## 3          1
## 4          1
## 5          1
## 6          0
```

```
# you can see that there is a column has presence-absence records (name of column is important to know:

#--- we can plot presence and absence points separately with different colours:
```

```r
plot(species[species$Occurrence == 1,],col='blue',pch=16)

points(species[species$Occurrence == 0,],col='red',pch=16)
```



```r
##########
# Let's read predictor variables (raster datasets)
# We have four Ascii-Grids, so, let's just take the name of all files ending to '.asc' to be able to re

path <- system.file("external", package="sdm") # path to the folder contains the data
lst <- list.files(path=path,pattern='asc$',full.names = T) # list the name of files in the specified pa

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

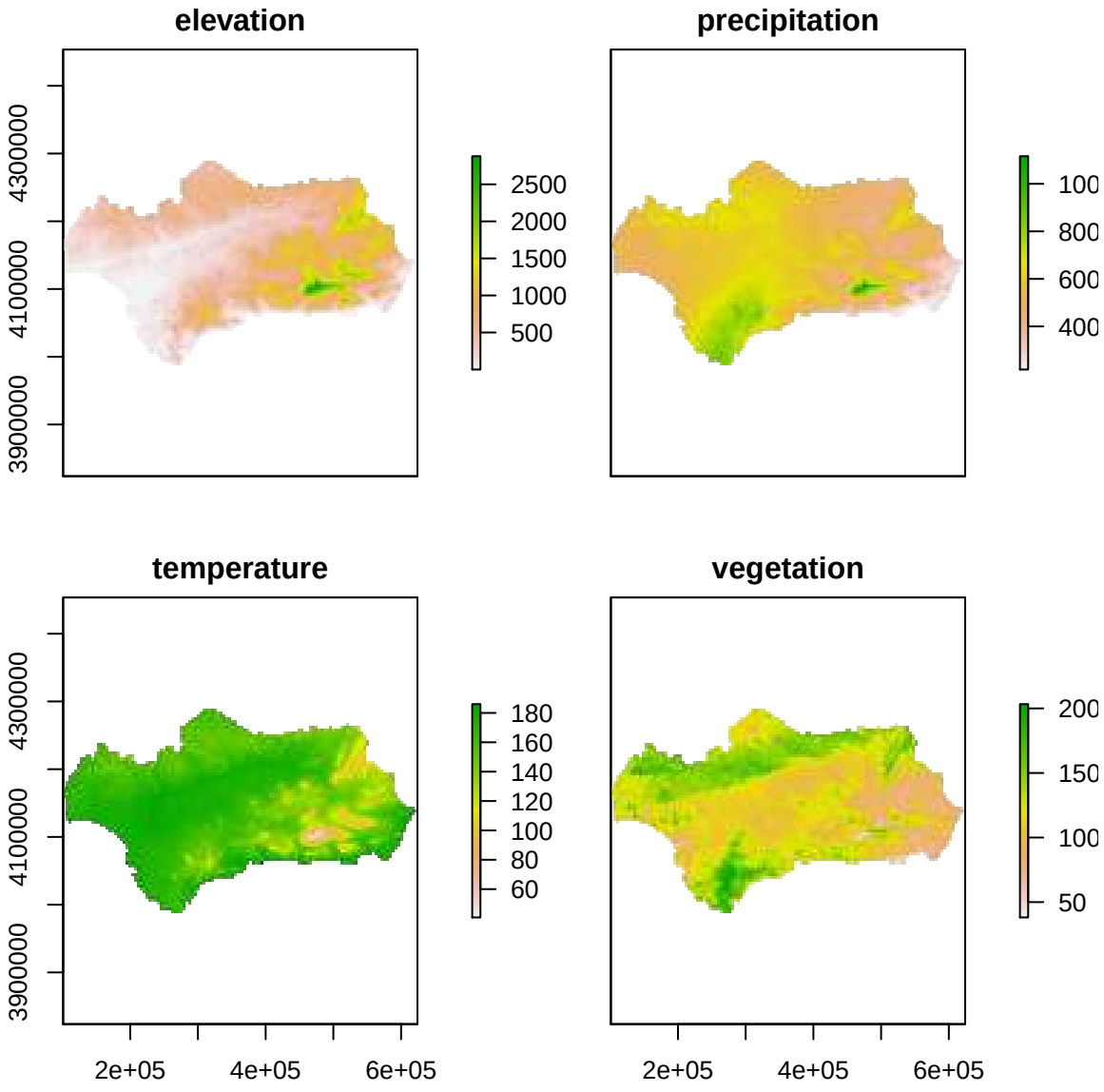
```
## [1] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/sdm/external/elevation.asc"
## [2] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/sdm/external/precipitation.asc"
## [3] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/sdm/external/temperature.asc"
## [4] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/sdm/external/vegetation.asc"
```

```r
# stack is a function in the raster package, to read/create a multi-layers raster dataset
preds <- stack(lst) # making a raster object

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

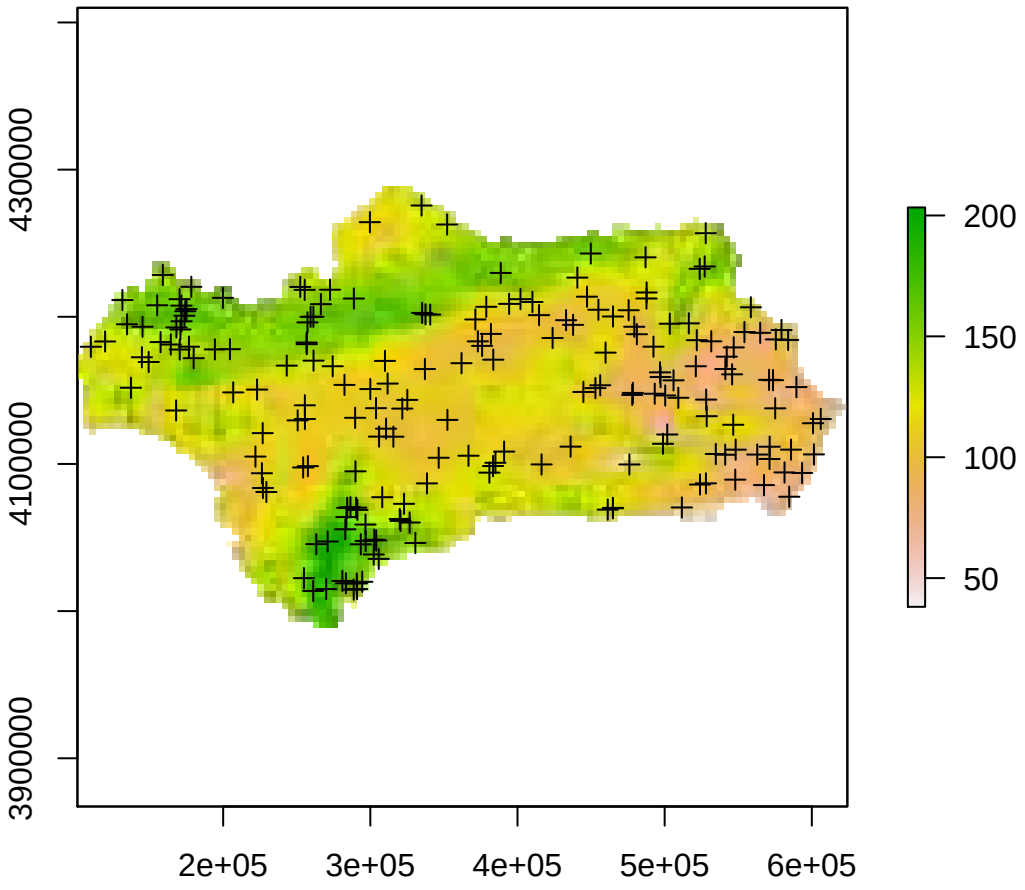
```
## class       : RasterStack
## dimensions  : 71, 124, 8804, 4  (nrow, ncol, ncell, nlayers)
## resolution  : 4219.223, 4219.223  (x, y)
## extent      : 100975.3, 624159, 3988830, 4288395  (xmin, xmax, ymin, ymax)
## coord. ref. : NA
## names       : elevation, precipitation, temperature, vegetation
```

```r
plot(preds)
```




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

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



Data preparation

So far, we used other packages to just read the data we need to use in our study. Now, we can use the sdm package. First, we need to read and put data in the package that creates a data object. This is very simple and efficient using the function **sdmData**.

In this function, we can specify the train dataset (can be spatial points or simply a data.frame), and predictors (if available separately as a raster object). In addition, if there is an independent dataset available to be used for measuring model performance (evaluation/validation), we can provide it through the test arguement. A formula can also be used to specify the response and explanatory variables (and in case if you have data.frame as input which contains coordinates, you can specify the coordinates in the formula as well, + more information). If the formula is not specified, the function tries to detect the species and predictor variables.

```
library(sdm)

d <- sdmData(formula=Occurrence~., train=species, predictors=preds)

d
```

```
## class                                 : sdmdata
## ============================================================
## number of species                     :  1
## species names                         :  Occurrence
## number of features                    :  4
## feature names                         :  elevation, precipitation, temperature, ...
```

```
## type                                 :  Presence-Absence
## has independet test data?            :  FALSE
## number of records                    :  200
## has Coordinates?                     :  TRUE
```

```
# we didn't really need the formula in this example, as it would be easy for the function to guess whic
d <- sdmData(train=species, predictors=preds)

d
```

```
## class                                : sdmdata
## ============================================================
## number of species                    :  1
## species names                        :  Occurrence
## number of features                   :  4
## feature names                        :  elevation, precipitation, temperature, ...
## type                                 :  Presence-Absence
## has independet test data?            :  FALSE
## number of records                    :  200
## has Coordinates?                     :  TRUE
```

```
# However, formula makes it so flexible to handle the variables, specifally if there are several other

# You may also want to take part of variables:
d <- sdmData(formula=Occurrence~precipitation+temperature, train=species, predictors=preds)

d
```

```
## class                                : sdmdata
## ============================================================
## number of species                    :  1
## species names                        :  Occurrence
## number of features                   :  2
## feature names                        :  precipitation, temperature
## type                                 :  Presence-Absence
## has independet test data?            :  FALSE
## number of records                    :  200
## has Coordinates?                     :  TRUE
```

```
d <- sdmData(formula= ~., train=species, predictors=preds)


#---
```

Model Fitting and Evaluation

When you create the data object (d in the above example), you would be able to fit the models. To do so, you are going to use the function **sdm**. In this function, you can specify the variables and the type of features can be generated based on, through a formula. In addition, the name of methods can be specifies as well as settings.

```
# in the following example, we use 3 different methods to fit the models.

m1 <- sdm(Occurrence~.,data=d,methods=c('glm','bioclim','brt'))

m1
```

```
## class                                  : sdmModels
## =======================================================
## number of species                      :  1
## number of modelling methods            :  3
## names of modelling methods             :  glm, bioclim, brt
## -------------------------------------------
## model run success percentage (per species)  :
## -------------------------------------------
## method          Occurrence
## ----------------------
## glm        :        100   %
## bioclim    :        100   %
## brt        :        100   %
##
## ########################################################################
## model performance (per species), using training test dataset:
## ------------------------------------------------------------------------------------
##
##  ## species   :  Occurrence
## ==========================
##
## methods    :     AUC     |     COR     |     TSS     |     Deviance
## ------------------------------------------------------------------------
## glm        :     0.88    |     0.7     |     0.69    |     0.83
## bioclim    :     0.77    |     0.47    |     0.43    |     1.64
## brt        :     0.9     |     0.72    |     0.69    |     0.92
```

```
# as you can see, a report is generates shows how many percent of models were successful, and their per
#-------------------

# in the above example, the performance statistics were calculated based on the training dataset (the d

# Here we are going to fit 4 models and evaluate them through 2 runs of subsampling, each draw 30 perce

m2 <- sdm(Occurrence~.,data=d,methods=c('rf','brt','bioclim','glm'),replicatin='sub',test.percent=30,n=

m2
```

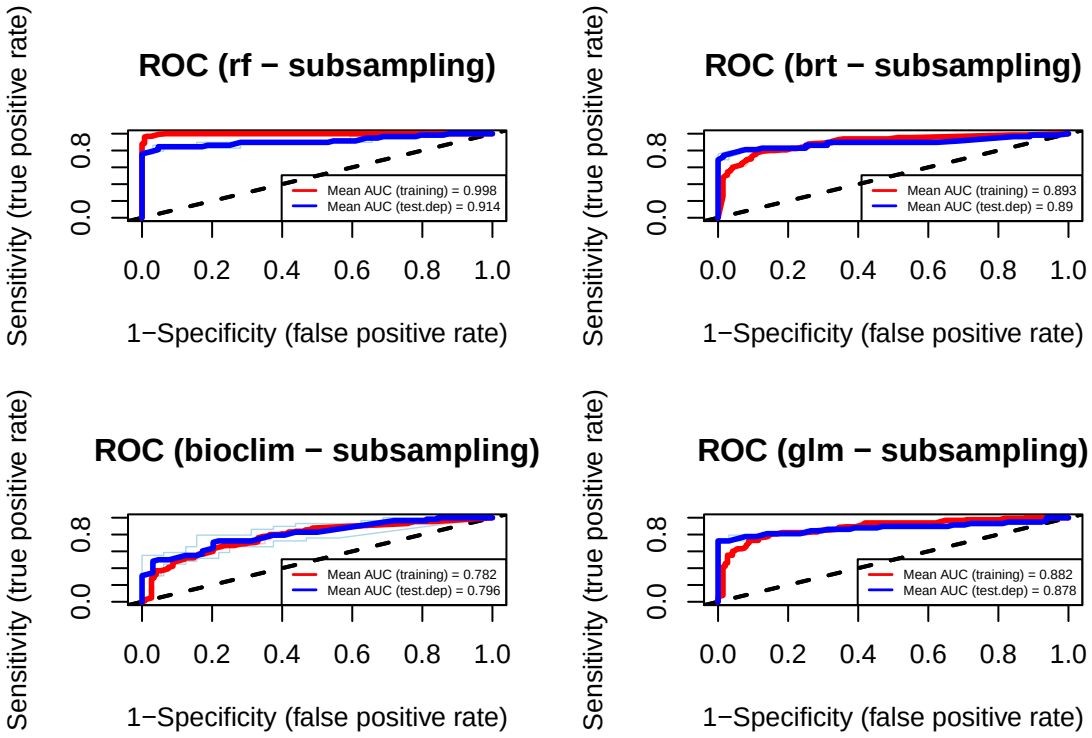
```
## class                                  : sdmModels
## =======================================================
## number of species                      :  1
## number of modelling methods            :  4
## names of modelling methods             :  rf, brt, bioclim, glm
## replicate.methods (data partitioning) :  subsampling
## number of replicates (each method)     :  2
## toral number of replicates per model   :  2 (per species)
## test percentage (in subsampling)       :  30
## -------------------------------------------
## model run success percentage (per species)  :
## -------------------------------------------
## method          Occurrence
## ----------------------
## rf         :        100   %
## brt        :        100   %
## bioclim    :        100   %
```

```
## glm        :        100   %
##
## ########################################################################
## model Mean performance (per species), using test dataset (generated using partitioning):
## ------------------------------------------------------------------------
##
##  ## species   :  Occurrence
## =========================
##
## methods    :     AUC     |     COR     |     TSS     |     Deviance
## ------------------------------------------------------------------------
## rf         :     0.91    |     0.76    |     0.8     |     0.75
## brt        :     0.89    |     0.75    |     0.75    |     0.92
## bioclim    :     0.8     |     0.53    |     0.52    |     1.59
## glm        :     0.88    |     0.72    |     0.74    |     0.82
```

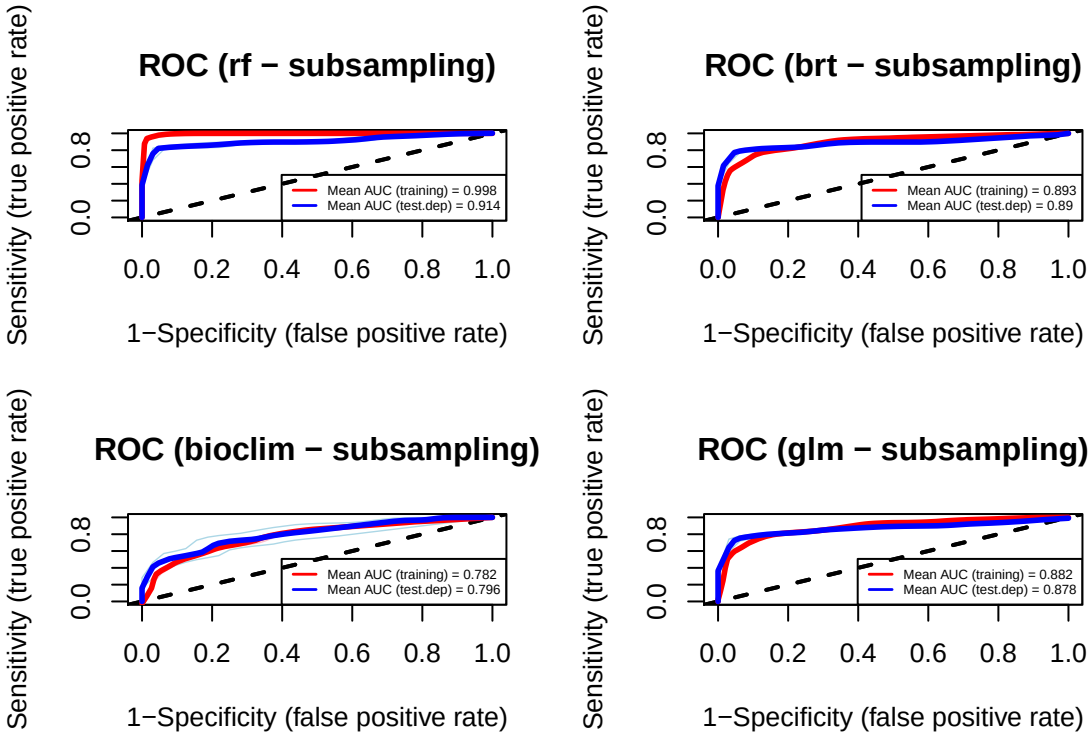
```
getModelInfo(m2) # info on runs including modelID, whether they are successfully fitted and evaluated,
```

```
##   modelID    species  method replication replicationID success training
## 1       1 Occurrence      rf subsampling             1    TRUE     TRUE
## 2       2 Occurrence      rf subsampling             2    TRUE     TRUE
## 3       3 Occurrence     brt subsampling             1    TRUE     TRUE
## 4       4 Occurrence     brt subsampling             2    TRUE     TRUE
## 5       5 Occurrence bioclim subsampling             1    TRUE     TRUE
## 6       6 Occurrence bioclim subsampling             2    TRUE     TRUE
## 7       7 Occurrence     glm subsampling             1    TRUE     TRUE
## 8       8 Occurrence     glm subsampling             2    TRUE     TRUE
##   test.dep test.indep
## 1     TRUE      FALSE
## 2     TRUE      FALSE
## 3     TRUE      FALSE
## 4     TRUE      FALSE
## 5     TRUE      FALSE
## 6     TRUE      FALSE
## 7     TRUE      FALSE
## 8     TRUE      FALSE
```

```
# We can generate the roc curve and compare the results for all models:
roc(m2)
```




```
# the plots can be smoothes:
roc(m2,smooth=TRUE)
```



Prediction

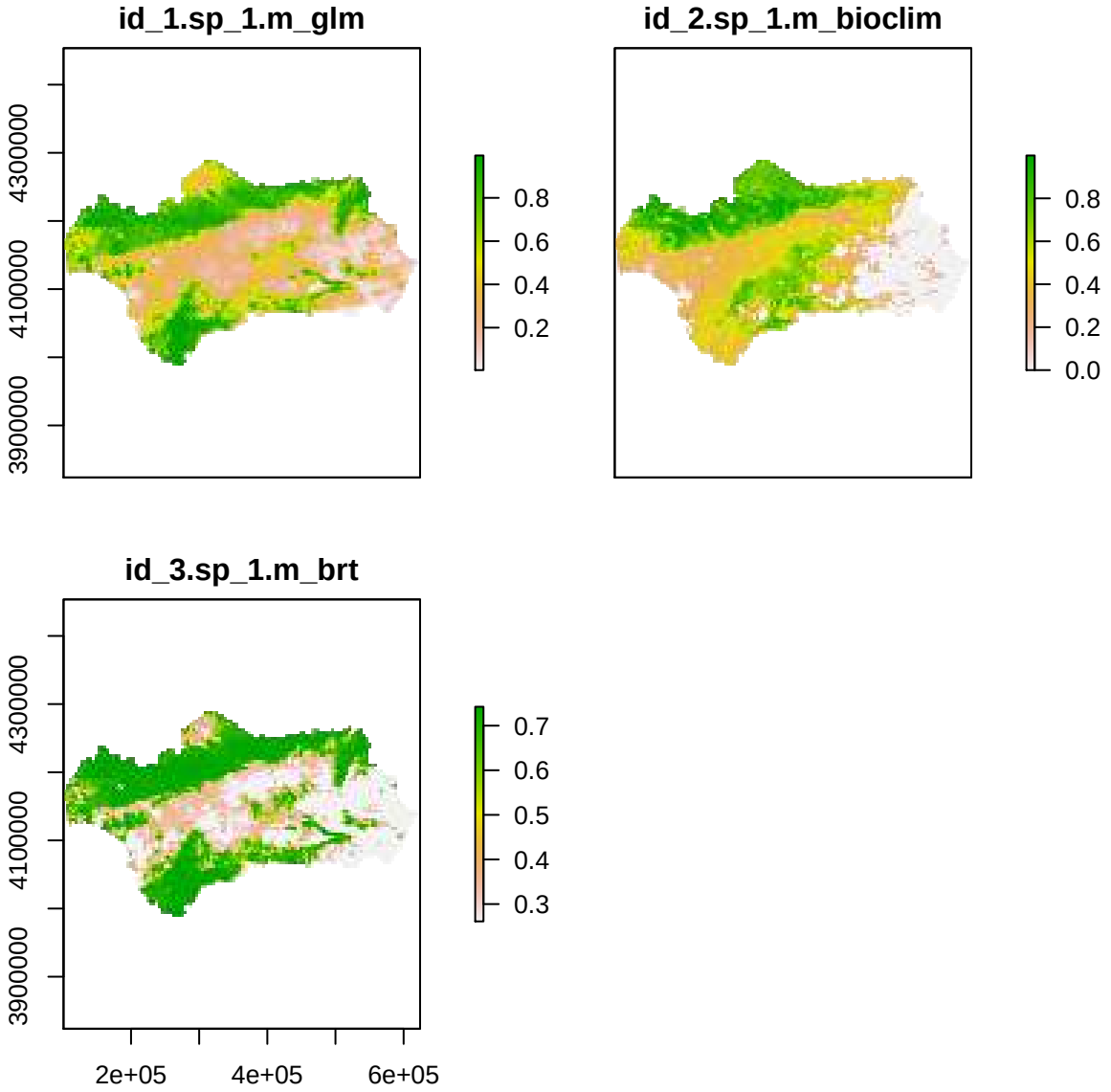
We can use the output of fitting, to predict into the study area, or project into a new location or a new time.

The predict function can be used for this purpose:

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

p1 <- predict(m1,newdata=preds,filename='p1.img') # many commonly used raster format is supported (thro

plot(p1)
```




```
p2 <- predict(m2,newdata=preds,filename='p2.img')

p2
```

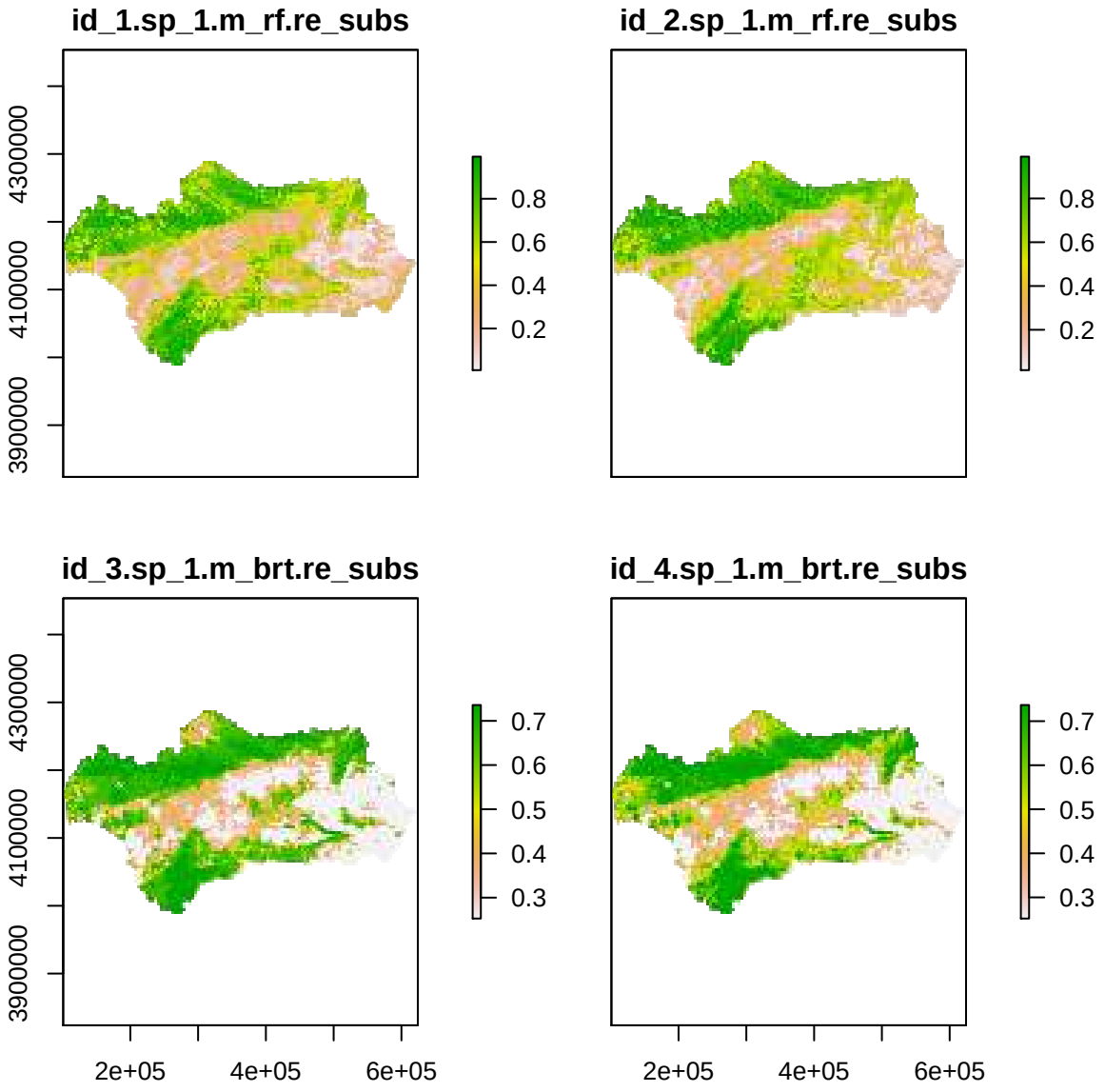
```
## class       : RasterBrick
## dimensions  : 71, 124, 8804, 8  (nrow, ncol, ncell, nlayers)
## resolution  : 4219.223, 4219.223  (x, y)
## extent      : 100975.3, 624159, 3988830, 4288395  (xmin, xmax, ymin, ymax)
## coord. ref. : NA
## data source : /Users/bnaimi/Dropbox/R_Books_Docs/r-gis.net/_sdm_rforge/sdm/pkg/sdm/vignettes/p2.img
```

```
## names       : id_1.sp_1.m_rf.re_subs, id_2.sp_1.m_rf.re_subs, id_3.sp_1.m_brt.re_subs, id_4.sp_1.m_b
## fullname    : id_1-species_Occurrence-method_rf-replication_subsampling, id_2-species_Occurrence-met
```

```
nlayers(p2)
```

```
## [1] 8
```

```
plot(p2[[1:4]]) # plot the first 12 rasters
```



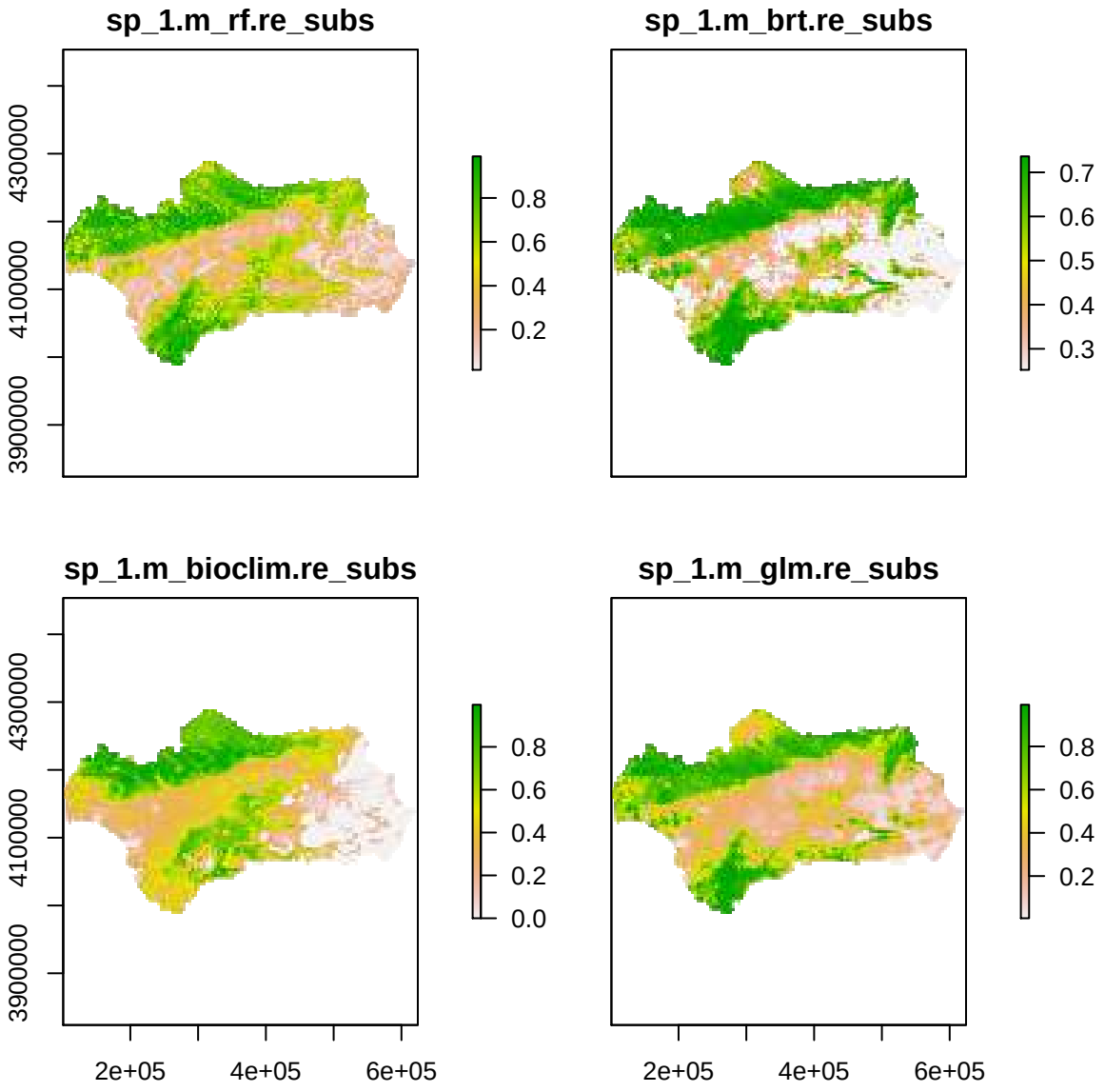

```
# we can take the mean of raster over different runs for each method and species:
p2m <- predict(m2,newdata=preds,filename='p2m.img',mean=T)

p2m
```

```
## class       : RasterBrick
## dimensions  : 71, 124, 8804, 4  (nrow, ncol, ncell, nlayers)
## resolution  : 4219.223, 4219.223  (x, y)
## extent      : 100975.3, 624159, 3988830, 4288395  (xmin, xmax, ymin, ymax)
## coord. ref. : NA
## data source : /Users/bnaimi/Dropbox/R_Books_Docs/r-gis.net/_sdm_rforge/sdm/pkg/sdm/vignettes/p2m.img
```

```
## names       : sp_1.m_rf.re_subs, sp_1.m_brt.re_subs, sp_1.m_bioclim.re_subs, sp_1.m_glm.re_subs
## min values  :       0.016933333,        0.252308882,             0.000000000,        0.004748883
## max values  :         0.9895667,          0.7362136,               0.9965962,          0.9929616
## fullname    : species_Occurrence-method_rf-replication (Mean)_subsampling, species_Occurrence-method
```

```
plot(p2m)
```




```
# full names of rasters:
getZ(p2m)
```

```
## [1] "species_Occurrence-method_rf-replication (Mean)_subsampling"
## [2] "species_Occurrence-method_brt-replication (Mean)_subsampling"
## [3] "species_Occurrence-method_bioclim-replication (Mean)_subsampling"
## [4] "species_Occurrence-method_glm-replication (Mean)_subsampling"
```

Ensemble forecasting

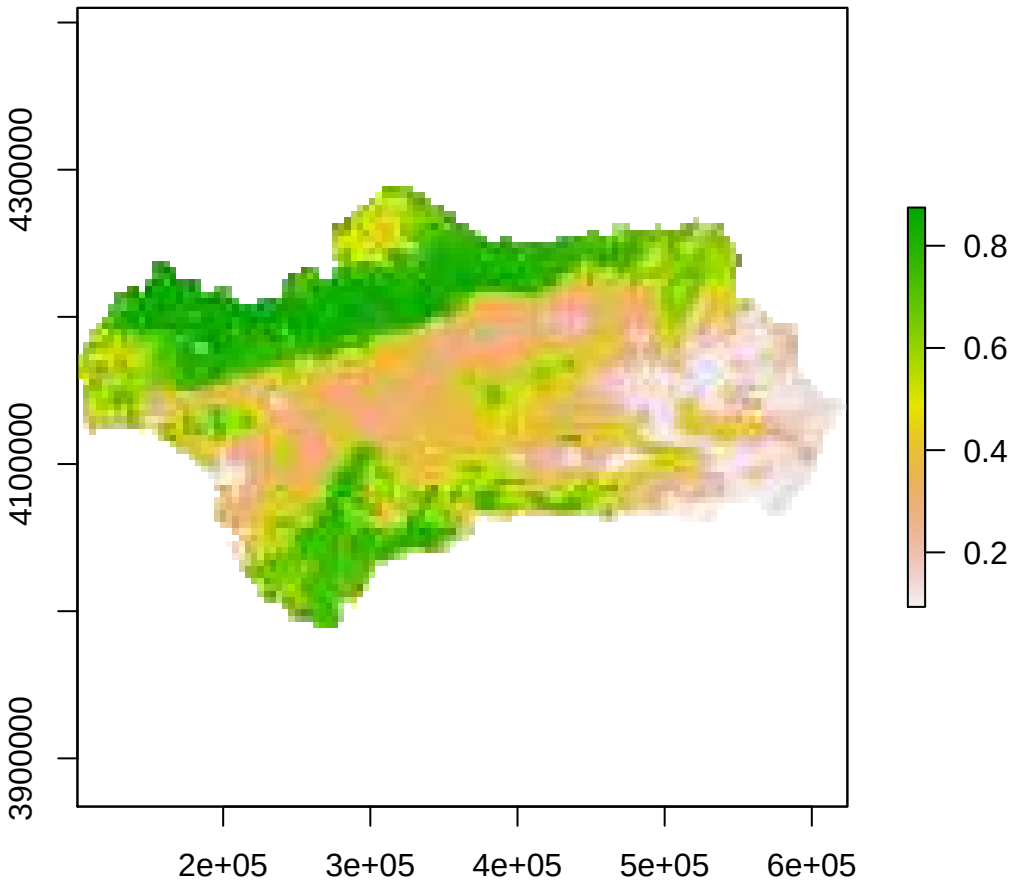
Studies have shown that predictions or projections by alternative models can be so variable that challenge the common practice of relying on one single method. A solution is to utilize several models ('ensembles') and use appropriate techniques to explore the resulting range of projections. Significant improvements on the robustness of a forecast can be achieved if an ensemble approach is used and the results analysed appropriately.

In the sdm package, the ensemble function can be used to generate an ensemble prediction or forecast based on the multiple models that are used in the sdm function. Several methods are implemented and can be used by a user in a flexible way. Here is an example:

```
# in the following, we predict the habitat suitability using the ensemble function
# since the newdata is a raster object, the output is also a raster object

# ensemble based on a Weighted averaging that is weighted using AUC statistic
e1 <- ensemble(m1,newdata=preds,filename='e1.img',setting=list(method='weighted',stat='AUC'))

plot(e1)
```



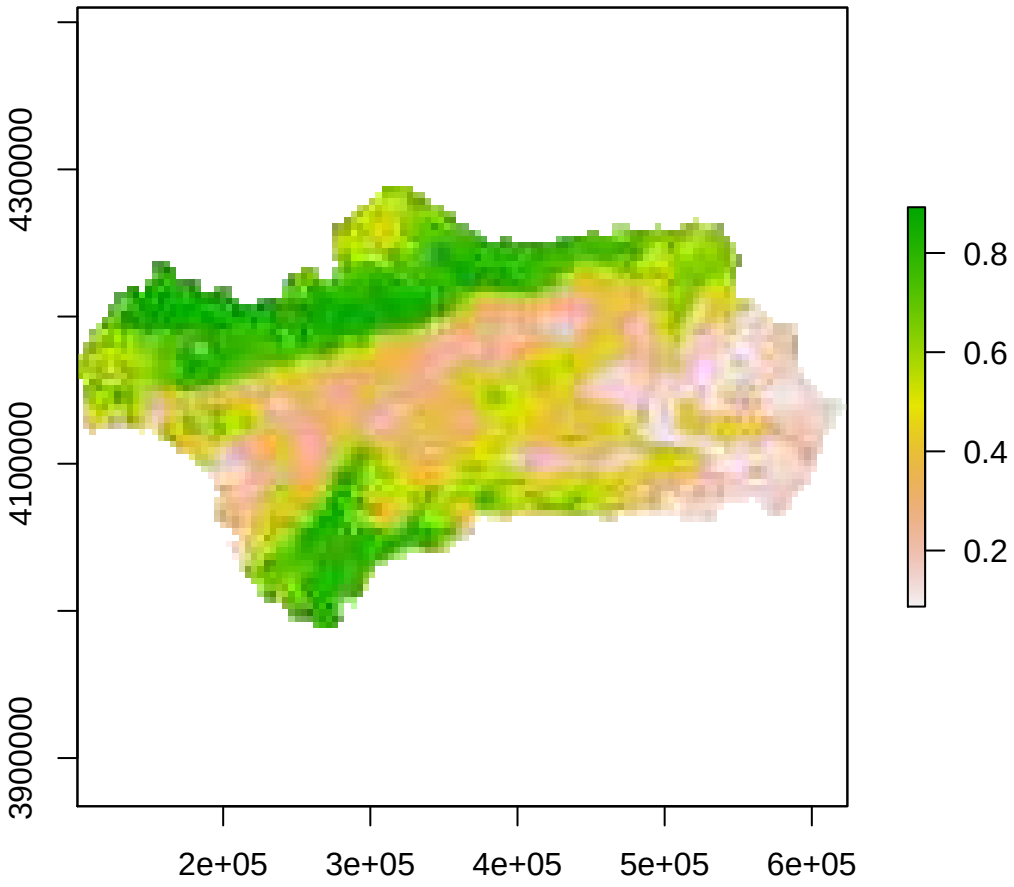

```
# ensemble based on a Weighted averaging that is weighted using TSS statistic with threshold criterion
e2 <- ensemble(m2,newdata=preds,filename='e2.img',setting=list(method='weighted',stat='TSS',opt=2))

e2
```

```
## class       : RasterLayer
## dimensions  : 71, 124, 8804  (nrow, ncol, ncell)
## resolution  : 4219.223, 4219.223  (x, y)
```

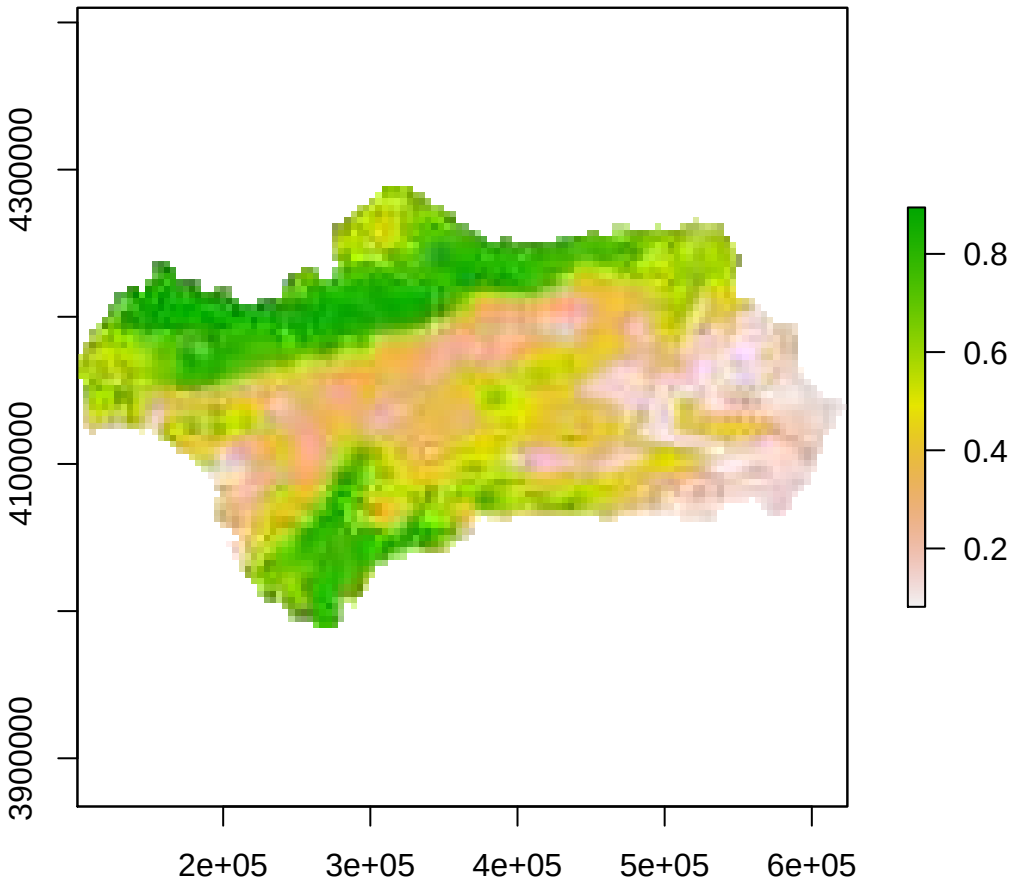
```
## extent      : 100975.3, 624159, 3988830, 4288395  (xmin, xmax, ymin, ymax)
## coord. ref. : NA
## data source : /Users/bnaimi/Dropbox/R_Books_Docs/r-gis.net/_sdm_rforge/sdm/pkg/sdm/vignettes/e2.img
## names       : e2
## values      : 0.08697034, 0.8923322  (min, max)
```

```
plot(e2)
```




```
# ensemble based on an Unweighted averaging
e3 <- ensemble(m2,newdata=preds,filename='e3.img',setting=list(method='unweighted'))

plot(e3)
```



there are other options in the setting argument that can specified by a used, for example, one may define a numeric verctor as weight, or specify the id of some models that should be incorporated into the ensemble procedure.

Reference

Naimi, B., Araujo, M.B. (2016) sdm: a reproducible and extensible R platform for species distribution modelling, Ecography, 39:368-375, DOI: 10.1111/ecog.01881