Spatial Distribution Models

Robert J. Hijmans and Jane Elith

May 20, 2021

CONTENTS

Robert J. Hijmans and Jane Elith

CHAPTER

ONE

INTRODUCTION

This document provides an introduction to species distribution modeling with *R*. Species distribution modeling (SDM) is also known under other names including climate envelope-modeling, habitat modeling, and (environmental or ecological) niche-modeling. The aim of SDM is to estimate the similarity of the conditions at any site to the conditions at the locations of known occurrence (and perhaps of non-occurrence) of a phenomenon. A common application of this method is to predict species ranges with climate data as predictors.

In SDM, the following steps are usually taken: (1) locations of occurrence of a species (or other phenomenon) are compiled; (2) values of environmental predictor variables (such as climate) at these locations are extracted from spatial databases; (3) the environmental values are used to fit a model to estimate similarity to the sites of occurrence, or another measure such as abundance of the species; (4) The model is used to predict the variable of interest across an the region of interest (and perhaps for a future or past climate).

We assume that you are familiar with most of the concepts in SDM. If in doubt, you could consult, for example, the book by Janet Franklin (2009), the somewhat more theoretical book by Peterson *et al.* (2011), or the recent review article by Elith and Leathwick (2009). It is important to have a good understanding of the interplay of environmental (niche) and geographic (biotope) space – see Colwell and Rangel (2009) and Peterson *et al.* (2011) for a discussion. SDM is a widely used approach but there is much debate on when and how to best use this method. While we refer to some of these issues, in this document we do not provide an in-depth discussion of this scientific debate. Rather, our objective is to provide practical guidance to implemeting the basic steps of SDM. We leave it to you to use other sources to determine the appropriate methods for your research; and to use the ample opportunities provided by the *R* environment to improve existing approaches and to develop new ones.

We also assume that you are already somewhat familiar with the *R* language and environment. It would be particularly useful if you already had some experience with statistical model fitting (e.g. the `glm` function) and with spatial data handling as implemented in the packages `raster` and `sp`. To familiarize yourself with model fitting see, for instance, the Documentation section on the CRAN webpage and any introduction to *R* txt. For the `raster` package you can consult these pages.

When we present *R* code we will provide some explanation if we think it might be difficult or confusing. We will do more of this earlier on in this document, so if you are relatively inexperienced with *R* and would like to ease into it, read this text in the presented order.

SDM have been implemented in *R* in many different ways. Here we focus on the functions in the `dismo` and the `raster` packages (but we also refer to other packages). If you want to test, or build on, some of the examples presented here, make sure you have the latest versions of these packages, and their dependencies, installed. If you are using a recent version of *R*, you can do that with:

```
install.packages(c('raster', 'rgdal', 'dismo', 'rJava'))
```

This document consists of 4 main parts. Part I is concerned with data preparation. This is often the most time consuming part of a species distribution modeling project. You need to collect a sufficient number of occurrence records that document presence (and perhaps absence or abundance) of the species of interest. You also need to have accurate and relevant environmental data (predictor variables) at a sufficiently high spatial resolution. We first discuss some aspects of assembling and cleaning species records, followed by a discussion of aspects of choosing and using the

predictor variables. A particularly important concern in species distribution modeling is that the species occurrence data adequately represent the actual distribution of the species studied. For instance, the species should be correctly identified, the coordinates of the location data need to be accurate enough to allow the general species/environment to be established, and the sample unbiased, or accompanied by information on known biases such that these can be taken into account. Part II introduces the main steps in SDM: fitting a model, making a prediction, and evaluating the result. Part III introduces different modeling methods in more detail (profile methods, regression methods, machine learning methods, and geographic methods). In Part IV we discuss a number of applications (e.g. predicting the effect of climate change), and a number of more advanced topics.

This is a work in progress. Suggestions are welcomed.

Robert J. Hijmans and Jane Elith

CHAPTER

TWO

DATA PREPARATION

2.1 Species occurrence data

Importing occurrence data into *R* is easy. But collecting, georeferencing, and cross-checking coordinate data is tedious. Discussions about species distribution modeling often focus on comparing modeling methods, but if you are dealing with species with few and uncertain records, your focus probably ought to be on improving the quality of the occurrence data (Lobo, 2008). All methods do better if your occurrence data is unbiased and free of error (Graham *et al.*, 2007) and you have a relatively large number of records (Wisz *et al.*, 2008). While we'll show you some useful data preparation steps you can do in *R*, it is necessary to use additional tools as well. For example, QGIS, is a very useful program for interactive editing of point data sets.

2.2 Importing occurrence data

In most cases you will have a file with point locality data representing the known distribution of a species. Below is an example of using `read.table` to read records that are stored in a text file.

We are using an example file that is installed with the `dismo` package, and for that reason we use a complex way to construct the filename, but you can replace that with your own filename. (remember to use forward slashes in the path of filenames!). `system.file` inserts the file path to where dismo is installed. If you haven't used the `paste` function before, it's worth familiarizing yourself with it (type `?paste`in the command window).

```
# loads the dismo library
library(dismo)
file <- paste0(system.file(package="dismo"), "/ex/bradypus.csv")
# this is the file we will use:
file
## [1] "C:/soft/R/R-4.0.5/library/dismo/ex/bradypus.csv"
```

Now read it and inspect the values of the file

```
bradypus <- read.table(file,  header=TRUE,  sep=",")
# or do:  bradypus <- read.csv(file)
# first rows
head(bradypus)
##                species      lon      lat
## 1 Bradypus variegatus -65.4000 -10.3833
## 2 Bradypus variegatus -65.3833 -10.3833
## 3 Bradypus variegatus -65.1333 -16.8000
## 4 Bradypus variegatus -63.6667 -17.4500
## 5 Bradypus variegatus -63.8500 -17.4000
```

(continues on next page)

(continued from previous page)

```
## 6 Bradypus variegatus -64.4167 -16.0000
# we only need columns 2 and 3:
bradypus <- bradypus[,2:3]
head(bradypus)
##        lon      lat
## 1 -65.4000 -10.3833
## 2 -65.3833 -10.3833
## 3 -65.1333 -16.8000
## 4 -63.6667 -17.4500
## 5 -63.8500 -17.4000
## 6 -64.4167 -16.0000
```

You can also read such data directly out of Excel files or from a database (see e.g. the `RODBC` package). Because this is a csv (comma separated values) file, we could also have used the `read.csv` function. No matter how you do it, the objective is to get a matrix (or a `data.frame`) with at least 2 columns that hold the coordinates of the locations where a species was observed. Coordinates are typically expressed as longitude and latitude (i.e. angular), but they could also be Easting and Northing in UTM or another planar coordinate reference system (map projection). The convention used here is to organize the coordinates columns so that longitude is the first and latitude the second column (think x and y axes in a plot; longitude is x, latitude is y); they often are in the reverse order, leading to undesired results. In many cases you will have additional columns, e.g., a column to indicate the species if you are modeling multiple species; and a column to indicate whether this is a 'presence' or an 'absence' record (a much used convention is to code presence with a 1 and absence with a 0).

If you do not have any species distribution data you can get started by downloading data from the Global Biodiversity Inventory Facility (GBIF). In the dismo package there is a function `gbif` that you can use for this. The data used below were downloaded (and saved to a permanent data set for use in this vignette) using the `gbif` function like this:

```
acaule <- gbif("solanum", "acaule*", geo=FALSE)
```

If you want to understand the order of the arguments given here to `gbif` or find out what other arguments you can use with this function, check out the help file (remember you can't access help files if the library is not loaded), by typing: `?gbif` or `help(gbif)`. Note the use of the asterix in "acaule*" to not only request `Solanum acaule`, but also variations such as the full name,*Solanum acaule* Bitter, or subspecies such as *Solanum acaule* subsp. *aemulans*.

Many occurence records may not have geographic coordinates. In this case, out of the 1366 records that GBIF returned (January 2013), there were 1082 records with coordinates (this was 699 and 54 in March 2010, a tremendous improvement!)

```
# load the saved S. acaule data
data(acaule)

# how many rows and colums?
dim(acaule)
## [1] 1366   25

#select the records that have longitude and latitude data
colnames(acaule)
##  [1] "species"              "continent"            "country"
##  [4] "adm1"                 "adm2"                 "locality"
##  [7] "lat"                  "lon"                  "coordUncertaintyM"
## [10] "alt"                  "institution"          "collection"
## [13] "catalogNumber"        "basisOfRecord"        "collector"
## [16] "earliestDateCollected" "latestDateCollected"  "gbifNotes"
## [19] "downloadDate"         "maxElevationM"        "minElevationM"
```

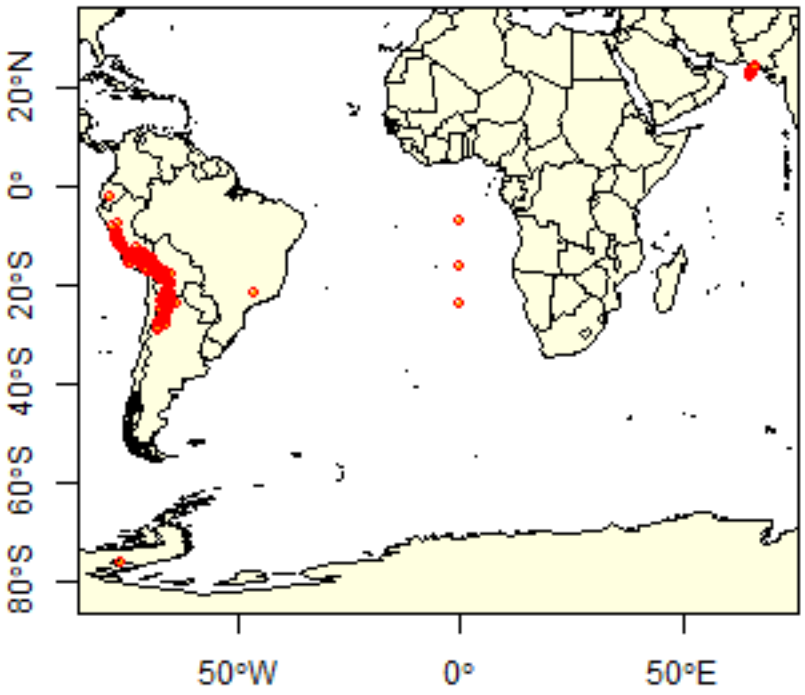
(continues on next page)

(continued from previous page)

```
## [22] "maxDepthM"             "minDepthM"             "ISO2"
## [25] "cloc"
acgeo <- subset(acaule, !is.na(lon) & !is.na(lat))
dim(acgeo)
## [1] 1082   25

# show some values
acgeo[1:4, c(1:5,7:10)]
##                   species     continent   country  adm1           adm2
## 1    Solanum acaule Bitter South America Argentina Jujuy Santa Catalina
## 2    Solanum acaule Bitter South America      Peru Cusco        Canchis
## 3 Solanum acaule f. acaule          <NA> Argentina  <NA>           <NA>
## 4 Solanum acaule f. acaule          <NA>   Bolivia  <NA>           <NA>
##        lat      lon coordUncertaintyM  alt
## 1 -21.9000 -66.1000              <NA>  NaN
## 2 -13.5000 -71.0000              <NA> 4500
## 3 -22.2666 -65.1333              <NA> 3800
## 4 -18.6333 -66.9500              <NA> 3700
```

Below is a simple way to make a map of the occurrence localities of *Solanum acaule*. It is important to make such maps to assure that the points are, at least roughly, in the right location.

```
library(maptools)
data(wrld_simpl)
plot(wrld_simpl, xlim=c(-80,70), ylim=c(-60,10), axes=TRUE, col="light yellow")
# restore the box around the map
box()

# add the points
points(acgeo$lon, acgeo$lat, col='orange', pch=20, cex=0.75)
# plot points again to add a border, for better visibility
points(acgeo$lon, acgeo$lat, col='red', cex=0.75)
```



The `wrld_simpl` dataset contains rough country outlines. You can use other datasets of polygons (or lines or points) as well. For example, you can download higher resolution data country and subnational administrative boundaries data with the `getData` function of the `raster` package. You can also read your own shapefile data into *R* using the `shapefile` function in the `raster` package.

2.3 Data cleaning

Data 'cleaning' is particularly important for data sourced from species distribution data warehouses such as GBIF. Such efforts do not specifically gather data for the purpose of species distribution modeling, so you need to understand the data and clean them appropriately, for your application. Here we provide an example.

`Solanum acaule` is a species that occurs in the higher parts of the Andes mountains of southern Peru, Bolivia and northern Argentina. Do you see any errors on the map?

There are a few records that map in the ocean just south of Pakistan. Any idea why that may have happened? It is a common mistake, missing minus signs. The coordinates are around (65.4, 23.4) but they should in Northern Argentina, around (-65.4, -23.4) (you can use the "click" function to query the coordintates on the map). There are two records (rows 303 and 885) that map to the same spot in Antarctica (-76.3, -76.3). The locality description says that is should be in Huarochiri, near Lima, Peru. So the longitude is probably correct, and erroneously copied to the latitude. Interestingly the record occurs twice. The orignal source is the International Potato Center, and a copy is provided by "SINGER" that aling the way appears to have "corrected" the country to Antarctica:

```
acaule[c(303,885),1:10]
##                           species continent    country adm1 adm2
## 303         solanum acaule acaule      <NA> Antarctica <NA> <NA>
## 885 solanum acaule acaule BITTER      <NA>       Peru <NA> <NA>
##                           locality   lat   lon coordUncertaintyM  alt
## 303                           <NA> -76.3 -76.3              <NA>  NaN
## 885 Lima P. Huarochiri Pacomanta -76.3 -76.3              <NA> 3800
```

The point in Brazil (record acaule[98,]) should be in soutern Bolivia, so this is probably due to a typo in the longitude. Likewise, there are also three records that have plausible latitudes, but longitudes that are clearly wrong, as they are in the Atlantic Ocean, south of West Africa. It looks like they have a longitude that is zero. In many data-bases you will find values that are 'zero' where 'no data' was intended. The `gbif` function (when using the default arguments) sets coordinates that are (0, 0) to `NA`, but not if one of the coordinates is zero. Let's see if we find them by searching for records with longitudes of zero.

Let's have a look at these records:

```
lonzero = subset(acgeo, lon==0)
# show all records, only the first 13 columns
lonzero[, 1:13]
##                                    species continent   country adm1 adm2
## 1159 Solanum acaule Bitter subsp. acaule      <NA> Argentina <NA> <NA>
## 1160 Solanum acaule Bitter subsp. acaule      <NA>   Bolivia <NA> <NA>
## 1161 Solanum acaule Bitter subsp. acaule      <NA>      Peru <NA> <NA>
## 1162 Solanum acaule Bitter subsp. acaule      <NA>      Peru <NA> <NA>
## 1163 Solanum acaule Bitter subsp. acaule      <NA> Argentina <NA> <NA>
## 1164 Solanum acaule Bitter subsp. acaule      <NA>   Bolivia <NA> <NA>
##                                                locality        lat lon
## 1159 between Quelbrada del Chorro and Laguna Colorada -23.716667   0
## 1160                                           Llave -16.083334   0
## 1161                    km 205 between Puno and Cuzco  -6.983333   0
## 1162                    km 205 between Puno and Cuzco  -6.983333   0
## 1163 between Quelbrada del Chorro and Laguna Colorada -23.716667   0
## 1164                                           Llave -16.083334   0
##      coordUncertaintyM  alt institution collection catalogNumber
## 1159              <NA> 3400         IPK         GB     WKS 30027
## 1160              <NA> 3900         IPK         GB     WKS 30050
## 1161              <NA> 4250         IPK  WKS 30048        304709
## 1162              <NA> 4250         IPK         GB     WKS 30048
## 1163              <NA> 3400         IPK  WKS 30027        304688
## 1164              <NA> 3900         IPK  WKS 30050        304711
```

The records are from Bolivia, Peru and Argentina, confirming that coordinates are in error. Alternatively, it could have been that the coordinates were correct, perhaps referring to a location in the Atlantic Ocean where a fish was caught rather than a place where *S. acaule* was collected). Records with the wrong species name can be among the hardest to correct (e.g., distinguishing between brown bears and sasquatch, Lozier *et al.*, 2009). The one record in Ecuador is like that, there is some debate whether that is actually a specimen of `S. albicans` or an anomalous hexaploid variety of *S. acaule*.

2.4 Duplicate records

Interestingly, another data quality issue is revealed above: each record in 'lonzero' occurs twice. This could happen because plant samples are often split and send to multiple herbariums. But in this case it seems that the IPK (The Leibniz Institute of Plant Genetics and Crop Plant Research) provided these data twice to the GBIF database (perhaps from seperate databases at IPK?). The function 'duplicated' can sometimes be used to remove duplicates.

```
# which records are duplicates (only for the first 10 columns)?
dups <- duplicated(lonzero[, 1:10])
# remove duplicates
lonzero  <-  lonzero[dups, ]
lonzero[,1:13]
##                                 species continent   country adm1 adm2
## 1162 Solanum acaule Bitter subsp. acaule      <NA>      Peru <NA> <NA>
## 1163 Solanum acaule Bitter subsp. acaule      <NA> Argentina <NA> <NA>
## 1164 Solanum acaule Bitter subsp. acaule      <NA>   Bolivia <NA> <NA>
##                                                locality        lat lon
## 1162                    km 205 between Puno and Cuzco  -6.983333   0
## 1163 between Quelbrada del Chorro and Laguna Colorada -23.716667   0
## 1164                                            Llave -16.083334   0
##      coordUncertaintyM  alt institution collection catalogNumber
## 1162              <NA> 4250         IPK         GB     WKS 30048
## 1163              <NA> 3400         IPK  WKS 30027        304688
## 1164              <NA> 3900         IPK  WKS 30050        304711
```

Another approach might be to detect duplicates for the same species and some coordinates in the data, even if the records were from collections by different people or in different years. (in our case, using species is redundant as we have data for only one species)

```
# differentiating by (sub) species
# dups2 <- duplicated(acgeo[, c('species', 'lon', 'lat')])
# ignoring (sub) species and other naming variation
dups2 <- duplicated(acgeo[, c('lon', 'lat')])
# number of duplicates
sum(dups2)
## [1] 483
# keep the records that are _not_ duplicated
acg <- acgeo[!dups2, ]
```

Let's repatriate the records near Pakistan to Argentina, and remove the records in Brazil, Antarctica, and with longitude=0

```
i <- acg$lon > 0 & acg$lat > 0
acg$lon[i] <- -1 * acg$lon[i]
acg$lat[i] <- -1 * acg$lat[i]
acg <- acg[acg$lon < -50 & acg$lat > -50, ]
```

2.5 Cross-checking

It is important to cross-check coordinates by visual and other means. One approach is to compare the country (and lower level administrative subdivisions) of the site as specified by the records, with the country implied by the coordinates (Hijmans *et al.*, 1999). In the example below we use the `coordinates` function from the `sp` package to create a `SpatialPointsDataFrame`, and then the `over` function, also from `sp`, to do a point-in-polygon query with the countries polygons.

We can make a SpatialPointsDataFrame using the statistical function notation (with a tilde):

```
library(sp)
coordinates(acg) <- ~lon+lat
crs(acg) <- crs(wrld_simpl)
class(acg)
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

We can now use the coordinates to do a spatial query of the polygons in wrld_simpl (a SpatialPolygonsDataFrame)

```
class(wrld_simpl)
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
ovr <- over(acg, wrld_simpl)
```

Object 'ov' has, for each point, the matching record from wrld_simpl. We need the variable 'NAME' in the data.frame of wrld_simpl

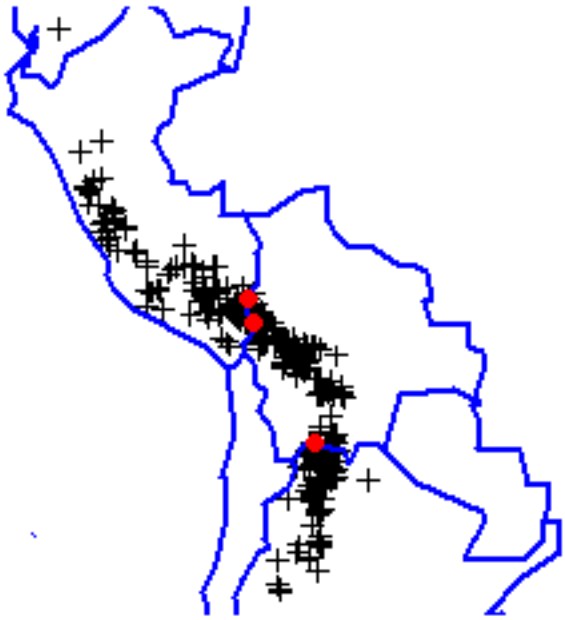
```
head(ovr)
##   FIPS ISO2 ISO3  UN      NAME   AREA  POP2005 REGION SUBREGION     LON     LAT
## 1   AR   AR  ARG  32 Argentina 273669 38747148     19         5 -65.167 -35.377
## 2   PE   PE  PER 604      Peru 128000 27274266     19         5 -75.552  -9.326
## 3   AR   AR  ARG  32 Argentina 273669 38747148     19         5 -65.167 -35.377
## 4   BL   BO  BOL  68   Bolivia 108438  9182015     19         5 -64.671 -16.715
## 5   BL   BO  BOL  68   Bolivia 108438  9182015     19         5 -64.671 -16.715
## 6   BL   BO  BOL  68   Bolivia 108438  9182015     19         5 -64.671 -16.715
cntr <- ovr$NAME
```

We should ask these two questions: (1) Which points (identified by their record numbers) do not match any country (that is, they are in an ocean)? (There are none (because we already removed the points that mapped in the ocean)). (2) Which points have coordinates that are in a different country than listed in the 'country' field of the gbif record

```
i <- which(is.na(cntr))
i
## integer(0)
j <- which(cntr != acg$country)
# for the mismatches, bind the country names of the polygons and points
cbind(cntr, acg$country)[j,]
##      cntr
## [1,] "27"  "Argentina"
## [2,] "172" "Bolivia"
## [3,] "172" "Bolivia"
## [4,] "172" "Bolivia"
```

In this case the mismatch is probably because wrld_simpl is not very precise as the records map to locations very close to the border between Bolivia and its neighbors.

```
plot(acg)
plot(wrld_simpl, add=T, border='blue', lwd=2)
points(acg[j, ], col='red', pch=20, cex=2)
```



See the sp package for more information on the `over` function. The wrld_simpl polygons that we used in the example above are not very precise, and they probably should not be used in a real analysis. See GADM for more detailed administrative division files, or use the `getData` function from the raster package (e.g. `getData('gadm', country='BOL', level=0)` to get the national borders of Bolivia; and `getData('countries')` to get all country boundaries).

2.6 Georeferencing

If you have records with locality descriptions but no coordinates, you should consider georeferencing these. Not all the records can be georeferenced. Sometimes even the country is unknown (country==“UNK”). Here we select only records that do not have coordinates, but that do have a locality description.

```
georef <- subset(acaule, (is.na(lon) | is.na(lat)) & ! is.na(locality) )
dim(georef)
## [1] 131  25
georef[1:3,1:13]
##                          species continent country adm1 adm2
## 606 solanum acaule acaule BITTER      <NA> Bolivia <NA> <NA>
## 607 solanum acaule acaule BITTER      <NA>    Peru <NA> <NA>
## 618 solanum acaule acaule BITTER      <NA>    Peru <NA> <NA>
##                                                          locality lat
## 606 La Paz P. Franz Tamayo Viscachani 3 km from Huaylapuquio to Pelechuco  NA
## 607                                  Puno P. San Roman Near Tinco Palca  NA
## 618                                            Puno P. Lampa Saraccocha  NA
##     lon coordUncertaintyM  alt institution             collection
## 606  NA              <NA> 4000      PER001 CIP - Potato collection
## 607  NA              <NA> 4000      PER001 CIP - Potato collection
## 618  NA              <NA> 4100      PER001 CIP - Potato collection
##     catalogNumber
## 606    CIP-762165
## 607    CIP-761962
## 618    CIP-762376
```

For georeferencing, you can try to use the dismo package function `geocode` that sends requests to the Google API (this used to be simple, but these days you need to first get an “API_KEY” from Google). We demonstrate below, but its use is generally not recommended because for accurate georeferencing you need a detailed map interface, and ideally one that allows you to capture the uncertainty associated with each georeference (Wieczorek *et al.*, 2004).

Here is an example for one of the records with longitude = 0, using Google’s geocoding service. We put the function into a ‘try’ function, to assure elegant error handling if the computer is not connected to the Internet. Note that we use the “cloc” (concatenated locality) field.

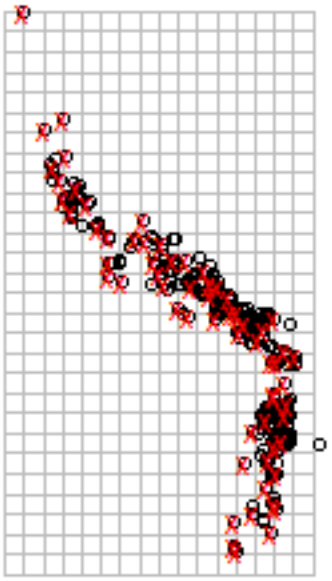
```
georef$cloc[4]
## [1] "Ayacucho P. Huamanga Minas Ckucho, Peru"
#b <- geocode(georef$cloc[4], geo_key="abcdef" )
#b
```

Before using the geocode function it is best to write the records to a table and “clean” them in a spreadsheet. Cleaning involves translation, expanding abbreviations, correcting misspellings, and making duplicates exactly the same so that they can be georeferenced only once. Then read the the table back into *R*, and create unique localities, georeference these and merge them with the original data.

2.7 Sampling bias

Sampling bias is frequently present in occurrence records (Hijmans *et al.*, 2001). One can attempt to remove some of the bias by subsampling records, and this is illustrated below. However, subsampling reduces the number of records, and it cannot correct the data for areas that have not been sampled at all. It also suffers from the problem that locally dense records might in fact be a true reflection of the relative suitable of habitat. As in many steps in SDM, you need to understand something about your data and species to implement them well. See Phillips *et al.* (2009) for an approach with MaxEnt to deal with bias in occurrence records for a group of species.

```
# create a RasterLayer with the extent of acgeo
r <- raster(acg)
# set the resolution of the cells to (for example) 1 degree
res(r) <- 1

# expand (extend) the extent of the RasterLayer a little
r <- extend(r, extent(r)+1)

# sample:
acsel <- gridSample(acg, r, n=1)

# to illustrate the method and show the result
p <- rasterToPolygons(r)
plot(p, border='gray')
points(acg)
# selected points in red
points(acsel, cex=1, col='red', pch='x')
```



Note that with the `gridSample` function you can also do 'chess-board' sampling. This can be useful to split the data in 'training' and 'testing' sets (see the model evaluation chapter).

At this point, it could be useful to save the cleaned data set. For example with the function `write.table` or `write.csv` so that we can use them later. We did that, and the saved file is available through dismo and can be retrieved like this:

```
file <- paste(system.file(package="dismo"), '/ex/acaule.csv', sep='')
acsel <- read.csv(file)
```

In a real research project you would want to spend much more time on this first data-cleaning and completion step, partly with *R*, but also with other programs.

##2.8 Exercises

1. Use the gbif function to download records for the African elephant (or another species of your preference, try to get one with between 10 and 100 records). Use option "geo=FALSE" to also get records with no (numerical) georeference.
2. Summarize the data: how many records are there, how many have coordinates, how many records without coordinates have a textual georeference (locality description)?
3. Use the 'geocode' function to georeference up to 10 records without coordinates
4. Make a simple map of all the records, using a color and symbol to distinguish between the coordinates from gbif and the ones returned by Google (via the geocode function). Use 'gmap' to create a basemap.

5. Do you think the observations are a reasonable representation of the distribution (and ecological niche) of the species?

More advanced:

6. Use the 'rasterize' function to create a raster of the number of observations and make a map. Use "wrld_simpl" from the maptools package for country boundaries.
7. Map the uncertainty associated with the georeferences. Some records in data returned by gbif have that. You can also extract it from the data returned by the geocode function.

CHAPTER

THREE

ABSENCE AND BACKGROUND POINTS

Some of the early species distribution model algorithms, such as Bioclim and Domain only use 'presence' data in the modeling process. Other methods also use 'absence' data or 'background' data. Logistic regression is the classical approach to analyzing presence and absence data (and it is still much used, often implemented in a generalized linear modeling (GLM) framework). If you have a large dataset with presence/absence from a well designed survey, you should use a method that can use these data (i.e. do not use a modeling method that only considers presence data). If you only have presence data, you can still use a method that needs absence data, by substituting absence data with background data.

Background data (e.g. Phillips *et al.* 2009) are not attempting to guess at absence locations, but rather to characterize environments in the study region. In this sense, background is the same, irrespective of where the species has been found. Background data establishes the environmental domain of the study, whilst presence data should establish under which conditions a species is more likely to be present than on average. A closely related but different concept, that of "pseudo-absences", is also used for generating the non-presence class for logistic models. In this case, researchers sometimes try to guess where absences might occur – they may sample the whole region except at presence locations, or they might sample at places unlikely to be suitable for the species. We prefer the background concept because it requires fewer assumptions and has some coherent statistical methods for dealing with the "overlap" between presence and background points (e.g. Ward et al. 2009; Phillips and Elith, 2011).

Survey-absence data has value. In conjunction with presence records, it establishes where surveys have been done, and the prevalence of the species given the survey effort. That information is lacking for presence-only data, a fact that can cause substantial difficulties for modeling presence-only data well. However, absence data can also be biased and incomplete, as discussed in the literature on detectability (e.g., Kéry *et al.*, 2010).

The `dismo` package has a function to sample random points (background data) from a study area. You can use a 'mask' to exclude area with no data `NA`, e.g. areas not on land. You can use an 'extent' to further restrict the area from which random locations are drawn.

In the example below, we first get the list of filenames with the predictor raster data (discussed in detail in the next chapter). We use a raster as a 'mask' in the `randomPoints` function such that the background points are from the same geographic area, and only for places where there are values (land, in our case).

Note that if the mask has the longitude/latitute coordinate reference system, function `randomPoints` selects cells according to cell area, which varies by latitude (as in Elith *et al.*, 2011)

```
library(dismo)
# get the file names
files <- list.files(path=paste(system.file(package="dismo"), '/ex',
                        sep=''),  pattern='grd',  full.names=TRUE )

# we use the first file to create a RasterLayer
mask <- raster(files[1])

# select 500 random points
```

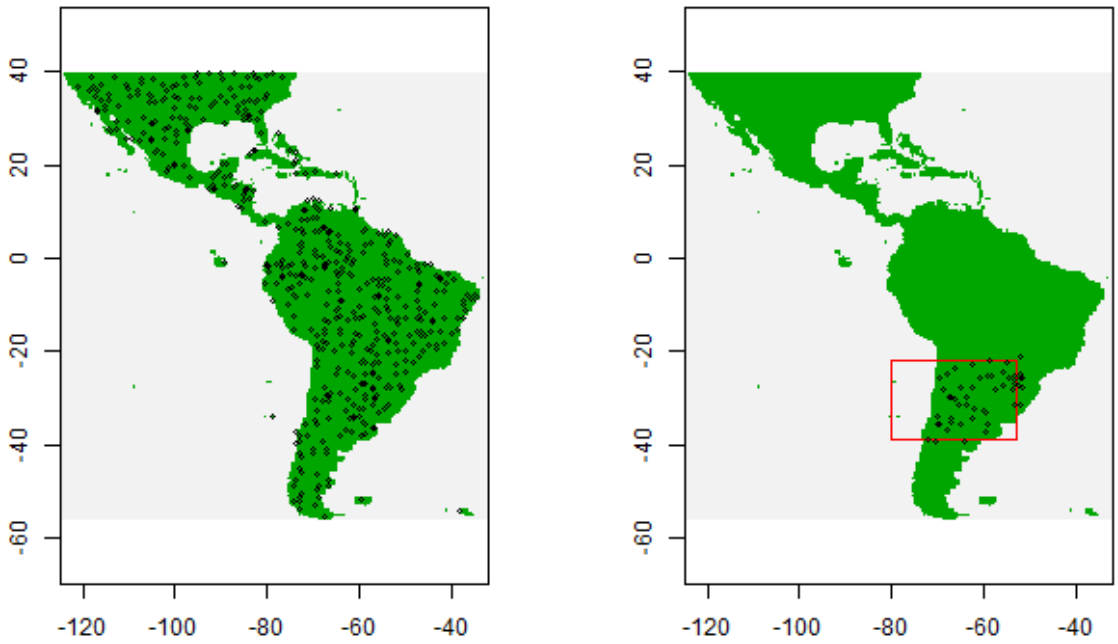
(continues on next page)

(continued from previous page)

```
# set seed to assure that the examples will always
# have the same random sample.
set.seed(1963)
bg <- randomPoints(mask, 500 )
```

And inspect the results by plotting

```
# set up the plotting area for two maps
par(mfrow=c(1,2))
plot(!is.na(mask), legend=FALSE)
points(bg, cex=0.5)

# now we repeat the sampling, but limit
# the area of sampling using a spatial extent
e <- extent(-80, -53, -39, -22)
bg2 <- randomPoints(mask, 50, ext=e)
plot(!is.na(mask), legend=FALSE)
plot(e, add=TRUE, col='red')
points(bg2, cex=0.5)
```



There are several approaches one could use to sample 'pseudo-absence' points, i.e., points from more restricted area than 'background'. VanDerWal et al. (2009) sampled withn a radius of presence points. Here is one way to implement that, using the `Solanum acaule` data.

We first read the cleaned and subsetted `S. acaule` data that we produced in the previous chapter from the csv file that comes with dismo:

```
file <- paste(system.file(package="dismo"), '/ex/acaule.csv', sep='')
ac <- read.csv(file)
```

ac is a data.frame. Let's change it into a SpatialPointsDataFrame

```
coordinates(ac) <- ~lon+lat
projection(ac) <- CRS('+proj=longlat +datum=WGS84')
```

We first create a 'circles' model (see the chapter about geographic models), using an arbitrary radius of 50 km

```
# circles with a radius of 50 km
x <- circles(ac, d=50000, lonlat=TRUE)
pol <- polygons(x)
```

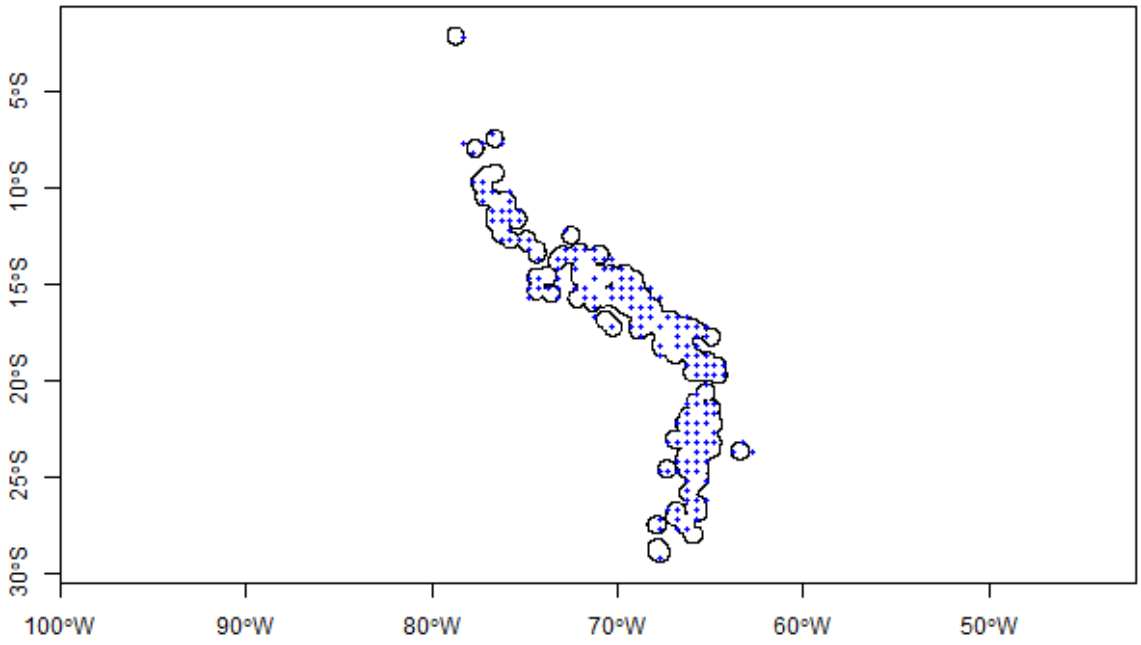
Note that you need to have the rgeos package installed for the circles function to 'dissolve' the circles (remove boundaries were circles overlap).

And then we take a random sample of points within the polygons. We only want one point per grid cell.

```
# sample randomly from all circles
samp1 <- spsample(pol, 250, type='random', iter=25)
## Warning in proj4string(obj): CRS object has comment, which is lost in output
# get unique cells
cells <- cellFromXY(mask, samp1)
length(cells)
## [1] 250
cells <- unique(cells)
length(cells)
## [1] 161
xy <- xyFromCell(mask, cells)
```

Plot to inspect the results:

```
plot(pol, axes=TRUE)
points(xy, cex=0.75, pch=20, col='blue')
```

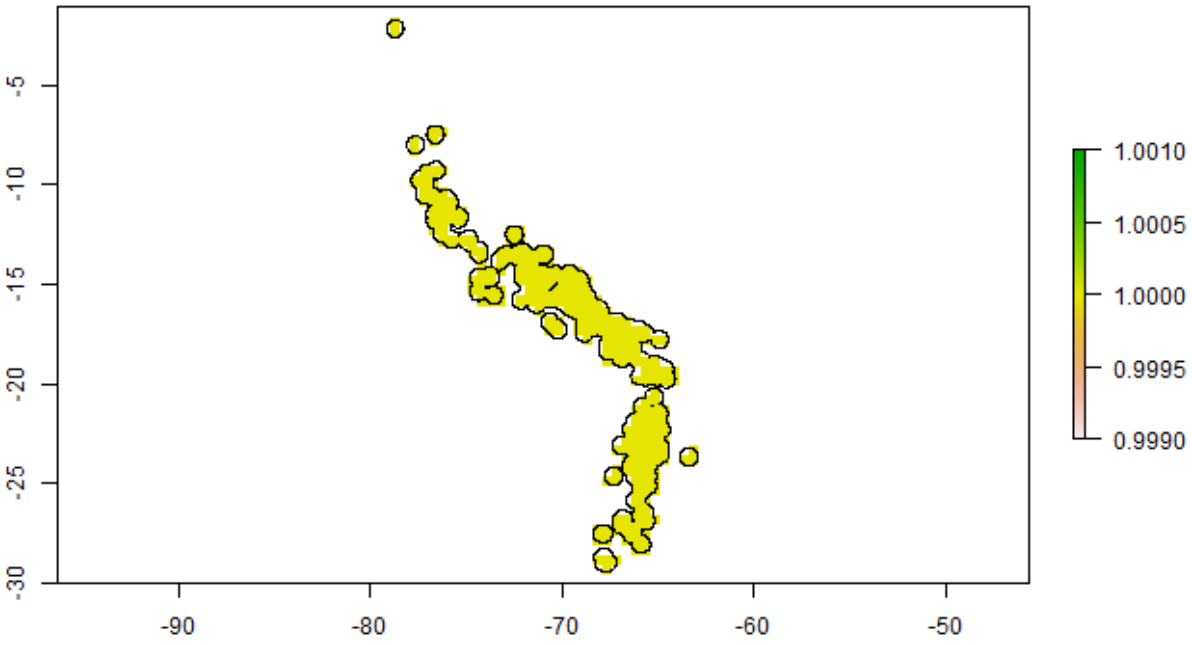


Note that the blue points are not all within the polygons (circles), as they now represent the centers of the selected cells from mask. We could choose to select only those cells that have their centers within the circles, using the overlay function.

```
spxy <- SpatialPoints(xy, proj4string=CRS('+proj=longlat +datum=WGS84'))
o <- over(spxy, geometry(x))
xyInside <- xy[!is.na(o), ]
```

Similar results could also be achieved via the raster functions `rasterize` or `extract`.

```
# extract cell numbers for the circles
v <- extract(mask, x@polygons, cellnumbers=T)
# use rbind to combine the elements in list v
v <- do.call(rbind, v)

# get unique cell numbers from which you could sample
v <- unique(v[,1])
head(v)
## [1] 15531 15717 17581 17582 17765 17767

# to display the results
m <- mask
m[] <- NA
m[v] <- 1
plot(m, ext=extent(x@polygons)+1)
plot(x@polygons, add=T)
```


-5
-10
-15
-20
-25
-30
-90
-80
-70
-60
-50
1.0010
1.0005
1.0000
0.9995
0.9990

CHAPTER

FOUR

ENVIRONMENTAL DATA

4.1 Raster data

In species distribution modeling, predictor variables are typically organized as raster (grid) type files. Each predictor should be a 'raster' representing a variable of interest. Variables can include climatic, soil, terrain, vegetation, land use, and other variables. These data are typically stored in files in some kind of GIS format. Almost all relevant formats can be used (including ESRI grid, geoTiff, netCDF, IDRISI). Avoid ASCII files if you can, as they tend to considerably slow down processing speed. For any particular study the layers should all have the same spatial extent, resolution, origin, and projection. If necessary, use functions like `crop`, `extend`, `aggregate`, `resample`, and `projectRaster` from the `raster` package. See the Introduction to spatial data manipulation for more information about function you can use to prepare your predictor variable data. See the help files and the vignette of the raster package for more info on how to do this.

The set of predictor variables (rasters) can be used to make a `RasterStack`, which is a collection of `RasterLayer' objects (see the Spatial Data tutorial for more info.

Here we make a list of files that are installed with the dismo package and then create a rasterStack from these, show the names of each layer, and finally plot them all.

First get the folder with our files. In this example it is the ("ex") directory of the dismo package. You do not need such a complex statement to get your own files (you just type the string).

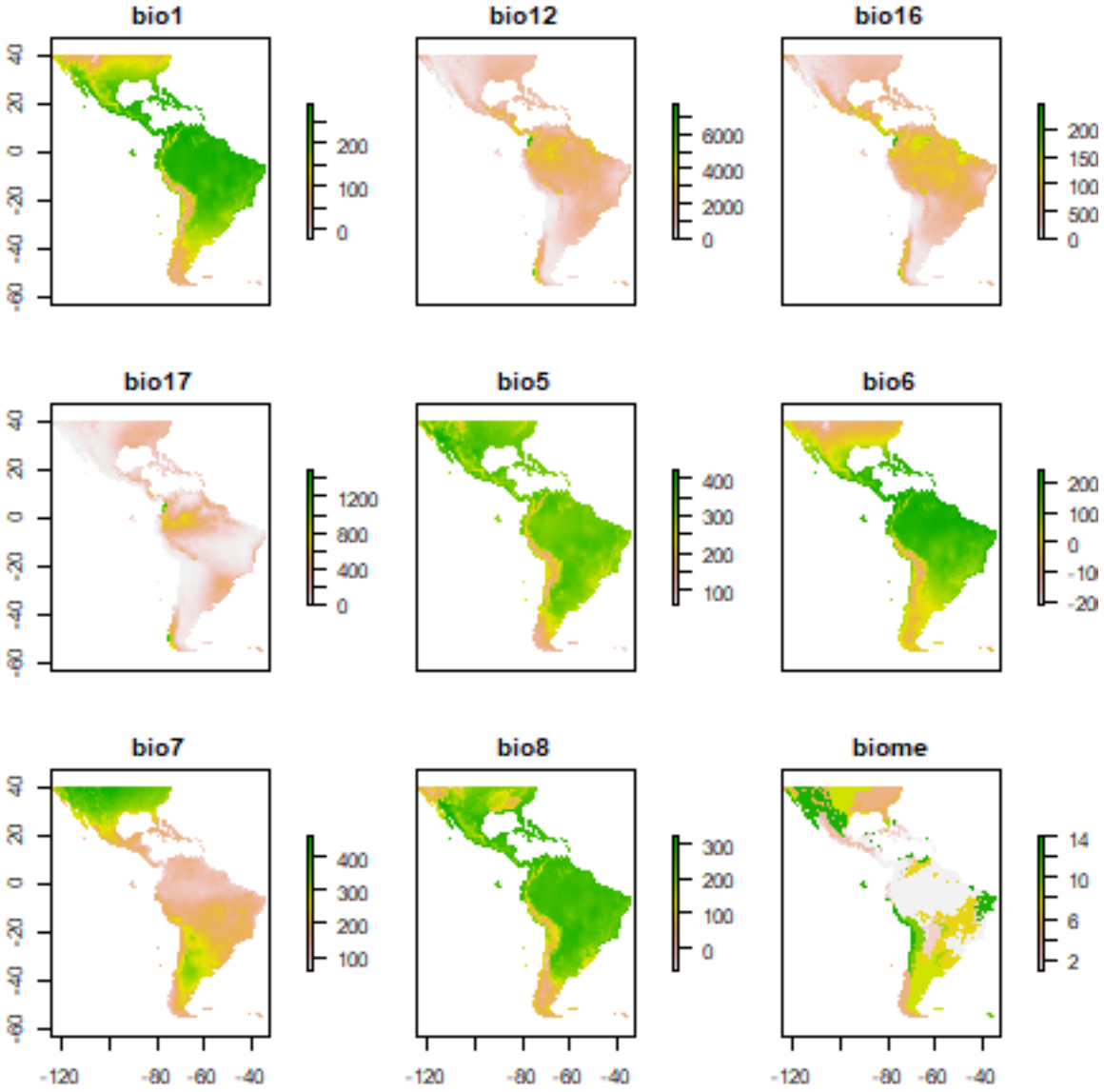
```
path <- file.path(system.file(package="dismo"), 'ex')
```

Now get the names of all the files with extension ".grd" in this folder. The $ sign indicates that the files must *end* with the characters 'grd'. By using `full.names=TRUE`, the full path names are retunred.

```
library(dismo)
files <- list.files(path, pattern='grd$', full.names=TRUE )
files
## [1] "C:/soft/R/R-4.0.5/library/dismo/ex/bio1.grd"
## [2] "C:/soft/R/R-4.0.5/library/dismo/ex/bio12.grd"
## [3] "C:/soft/R/R-4.0.5/library/dismo/ex/bio16.grd"
## [4] "C:/soft/R/R-4.0.5/library/dismo/ex/bio17.grd"
## [5] "C:/soft/R/R-4.0.5/library/dismo/ex/bio5.grd"
## [6] "C:/soft/R/R-4.0.5/library/dismo/ex/bio6.grd"
## [7] "C:/soft/R/R-4.0.5/library/dismo/ex/bio7.grd"
## [8] "C:/soft/R/R-4.0.5/library/dismo/ex/bio8.grd"
## [9] "C:/soft/R/R-4.0.5/library/dismo/ex/biome.grd"
```

Now create a RasterStack of predictor variables.

```
predictors <- stack(files)
predictors
## class      : RasterStack
## dimensions : 192, 186, 35712, 9  (nrow, ncol, ncell, nlayers)
## resolution : 0.5, 0.5  (x, y)
## extent     : -125, -32, -56, 40  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## names      : bio1, bio12, bio16, bio17, bio5, bio6, bio7, bio8, biome
## min values :  -23,     0,     0,     0,   61, -212,   60,  -66,     1
## max values :  289,  7682,  2458,  1496,  422,  242,  461,  323,    14
names(predictors)
## [1] "bio1"  "bio12" "bio16" "bio17" "bio5"  "bio6"  "bio7"  "bio8"  "biome"
plot(predictors)
```

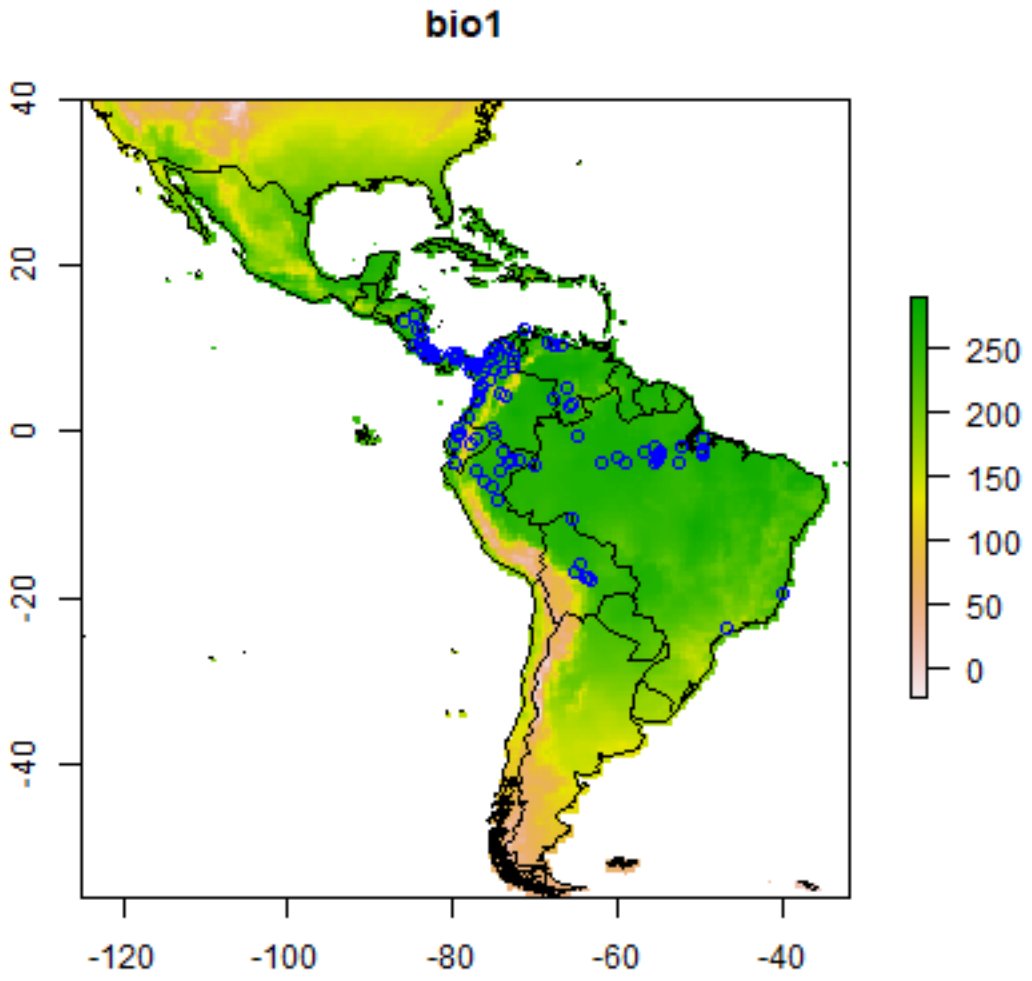


We can also make a plot of a single layer in a RasterStack, and plot some additional data on top of it. First get the world boundaries and the bradypus data:

```
library(maptools)
data(wrld_simpl)
file <- paste(system.file(package="dismo"), "/ex/bradypus.csv", sep="")
bradypus <- read.table(file,  header=TRUE,  sep=',')
# we do not need the first column
bradypus  <- bradypus[,-1]
```

And now plot:

```
# first layer of the RasterStack
plot(predictors, 1)

# note the "add=TRUE" argument with plot
plot(wrld_simpl, add=TRUE)
# with the points function, "add" is implicit
points(bradypus, col='blue')
```



The example above uses data representing 'bioclimatic variables' from the WorldClim database (Hijmans *et al.*, 2004) and 'terrestrial biome' data from the WWF (Olsen *et al.*, 2001). You can go to these websites if you want higher resolution data. You can also use the `getData` function from the `raster` package to download WorldClim climate data.

Predictor variable selection can be important, particularly if the objective of a study is explanation. See, e.g., Austin and Smith (1987), Austin (2002), Mellert *et al.*, (2011). The early applications of species modeling tended to focus on explanation (Elith and Leathwick 2009). Nowadays, the objective of SDM tends to be prediction. For prediction within the same geographic area, variable selection might arguably be relatively less important, but for many prediction tasks (e.g. to new times or places, see below) variable selection is critically important. In all cases it is important to use variables that are relevant to the ecology of the species (rather than with the first dataset that can be found on the web!). In some cases it can be useful to develop new, more ecologically relevant, predictor variables from existing data. For example, one could use land cover data and the `focal` function in the `raster` package to create a new variable that indicates how much forest area is available within x km of a grid cell, for a species that might have a home range of x.

4.2 Extracting values from rasters

We now have a set of predictor variables (rasters) and occurrence points. The next step is to extract the values of the predictors at the locations of the points. (This step can be skipped for the modeling methods that are implemented in the dismo package). This is a very straightforward thing to do using the 'extract' function from the raster package. In the example below we use that function first for the `Bradypus` occurrence points, then for 500 random background points. We combine these into a single `data.frame` in which the first column (variable 'pb') indicates whether this is a presence or a background point. 'biome' is categorical variable (called a 'factor' in *R*) and it is important to explicitly define it that way, so that it won't be treated like any other numerical variable.

```
presvals <- extract(predictors, bradypus)
# setting random seed to always create the same
# random set of points for this example
set.seed(0)
backgr <- randomPoints(predictors, 500)
absvals <- extract(predictors, backgr)
pb <- c(rep(1, nrow(presvals)), rep(0, nrow(absvals)))
sdmdata <- data.frame(cbind(pb, rbind(presvals, absvals)))
sdmdata[,'biome'] = as.factor(sdmdata[,'biome'])
head(sdmdata)
##   pb bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8 biome
## 1  1  263  1639   724    62  338  191  147  261     1
## 2  1  263  1639   724    62  338  191  147  261     1
## 3  1  253  3624  1547   373  329  150  179  271     1
## 4  1  243  1693   775   186  318  150  168  264     1
## 5  1  243  1693   775   186  318  150  168  264     1
## 6  1  252  2501  1081   280  326  154  172  270     1
tail(sdmdata)
##     pb bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8 biome
## 611  0  265  1622   749    66  344  184  160  266     1
## 612  0  241  1648   475   354  294  183  112  229     1
## 613  0  154   731   250    80  319   18  301  198     8
## 614  0  169  1195   316   273  293   69  224  122     7
## 615  0  247  2965  1173   327  314  168  146  253     1
## 616  0  117   495   231    36  336  -96  432  216     8
summary(sdmdata)
##        pb              bio1           bio12            bio16
```

(continues on next page)

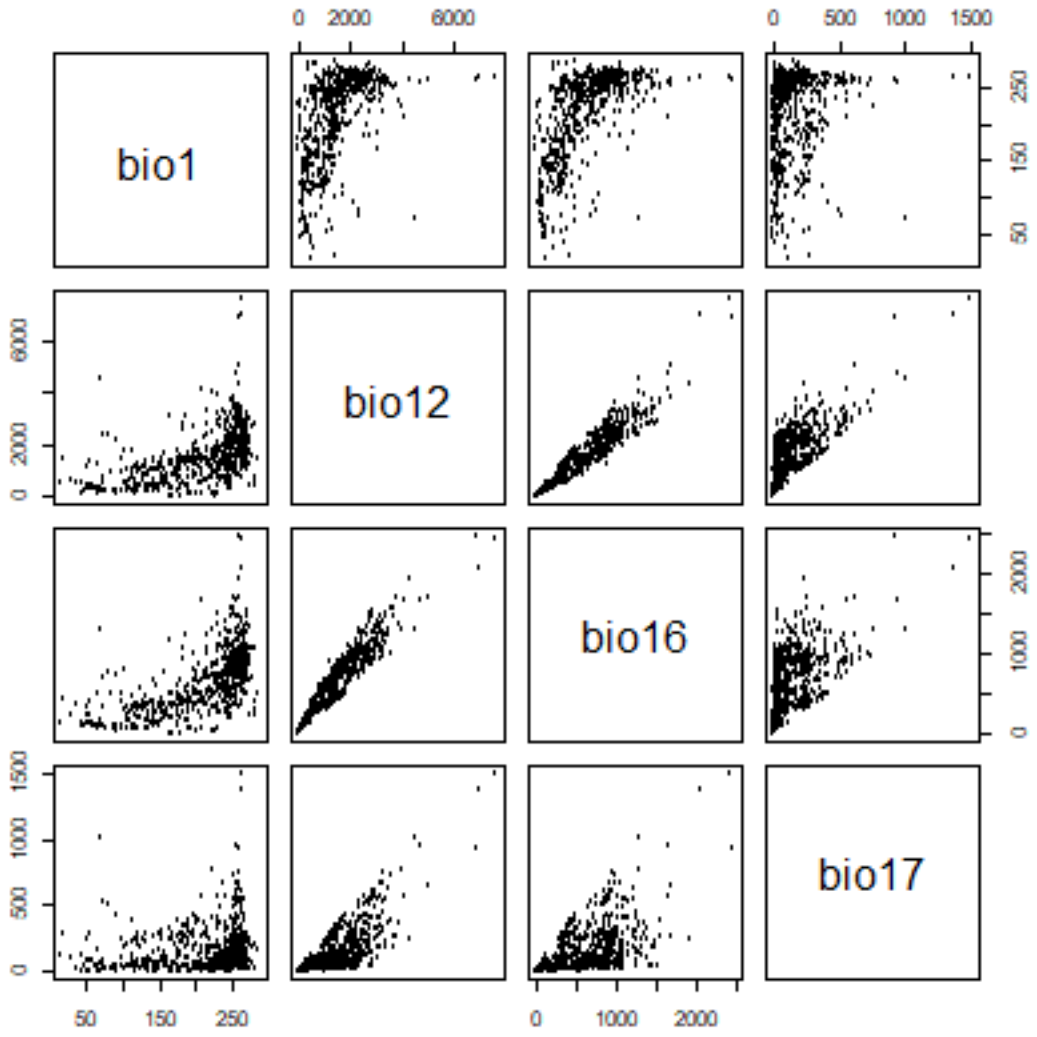
(continued from previous page)

```
##  Min.   :0.0000   Min.   : 16.0   Min.   :   0.0   Min.   :   0.0
##  1st Qu.:0.0000   1st Qu.:176.8   1st Qu.: 821.5   1st Qu.: 341.8
##  Median :0.0000   Median :242.0   Median :1497.0   Median : 645.5
##  Mean   :0.1883   Mean   :213.9   Mean   :1629.5   Mean   : 656.4
##  3rd Qu.:0.0000   3rd Qu.:261.0   3rd Qu.:2266.8   3rd Qu.: 917.0
##  Max.   :1.0000   Max.   :286.0   Max.   :7682.0   Max.   :2458.0
##
##      bio17            bio5            bio6            bio7
##  Min.   :   0.0   Min.   : 88.0   Min.   :-167.0   Min.   : 68.0
##  1st Qu.:  42.0   1st Qu.:303.0   1st Qu.:  45.0   1st Qu.:118.8
##  Median : 110.0   Median :320.0   Median : 151.0   Median :160.0
##  Mean   : 169.2   Mean   :309.3   Mean   : 119.3   Mean   :190.0
##  3rd Qu.: 240.5   3rd Qu.:333.0   3rd Qu.: 202.2   3rd Qu.:247.0
##  Max.   :1496.0   Max.   :410.0   Max.   : 232.0   Max.   :440.0
##
##       bio8           biome
##  Min.   :-20.0   1      :299
##  1st Qu.:215.0   7      : 65
##  Median :250.0   8      : 61
##  Mean   :225.1   13     : 51
##  3rd Qu.:262.0   2      : 49
##  Max.   :323.0   4      : 33
##                  (Other): 58
```

There are alternative approaches possible here. For example, one could extract multiple points in a radius as a potential means for dealing with mismatch between location accuracy and grid cell size. If one would make 10 datasets that represent 10 equally valid "samples" of the environment in that radius, that could be then used to fit 10 models and explore the effect of uncertainty in location.

To visually investigate colinearity in the environmental data (at the presence and background points) you can use a pairs plot. See Dormann *et al.* (2013) for a discussion of methods to remove colinearity.

```
pairs(sdmdata[,2:5], cex=0.1)
```



A pairs plot of the values of the climate data at the Bradypus occurrence sites.

To use the `sdmdata` and `presvals` in the next chapter, we save it to disk.

```
saveRDS(sdmdata, "sdm.Rds")
saveRDS(presvals, "pvals.Rds")
```

CHAPTER

FIVE

MODEL FITTING, PREDICTION, AND EVALUATION

5.1 Model fitting

Model fitting is technically quite similar across the modeling methods that exist in *R*. Most methods take a `formula` identifying the dependent and independent variables, accompanied with a `data.frame` that holds these variables. Details on specific methods are provided further down on this document, in part III.

A simple formula could look like: `y ~ x1 + x2 + x3`, i.e., y is a function of x1, x2, and x3. Another example is `y ~ .`, which means that y is a function of all other variables in the `data.frame` provided to the function. See `help('formula')` for more details about the formula syntax. In the example below, the function `glm` is used to fit generalized linear models. `glm` returns a model object.

Note that in the examples below, we are using the data.frame 'sdmdata' again. First read `sdmdata` from disk (we saved it at the end of the previous chapter).

```
library(dismo)
sdmdata <- readRDS("sdm.Rds")
presvals <- readRDS("pvals.Rds")
```

```
m1 <- glm(pb ~ bio1 + bio5 + bio12, data=sdmdata)
class(m1)
## [1] "glm" "lm"
summary(m1)
##
## Call:
## glm(formula = pb ~ bio1 + bio5 + bio12, data = sdmdata)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.95534  -0.23380  -0.09987   0.07831   0.88796
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.550e-01  1.093e-01   1.418 0.156593
## bio1         1.621e-03  3.960e-04   4.093 4.84e-05 ***
## bio5        -1.614e-03  4.725e-04  -3.417 0.000676 ***
## bio12        1.140e-04  1.676e-05   6.806 2.39e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1220413)
```

(continues on next page)

(continued from previous page)

```
##
##     Null deviance: 94.156  on 615  degrees of freedom
## Residual deviance: 74.689  on 612  degrees of freedom
## AIC: 458.43
##
## Number of Fisher Scoring iterations: 2

m2 = glm(pb ~ ., data=sdmdata)
m2
##
## Call:  glm(formula = pb ~ ., data = sdmdata)
##
## Coefficients:
## (Intercept)         bio1        bio12        bio16        bio17         bio5
##   0.2644272   -0.0020029    0.0004336   -0.0006207   -0.0007843    0.0035996
##        bio6         bio7         bio8       biome2       biome3       biome4
##  -0.0025153   -0.0041593    0.0008508   -0.0899869   -0.1262125   -0.0699228
##      biome5       biome7       biome8       biome9      biome10      biome11
##  -0.0136612   -0.1993102    0.0057666    0.0465949    0.0055868   -0.2705639
##     biome12      biome13      biome14
##  -0.0450194    0.0839402   -0.1689054
##
## Degrees of Freedom: 615 Total (i.e. Null);  595 Residual
## Null Deviance:       94.16
## Residual Deviance: 69.17     AIC: 445.1
```

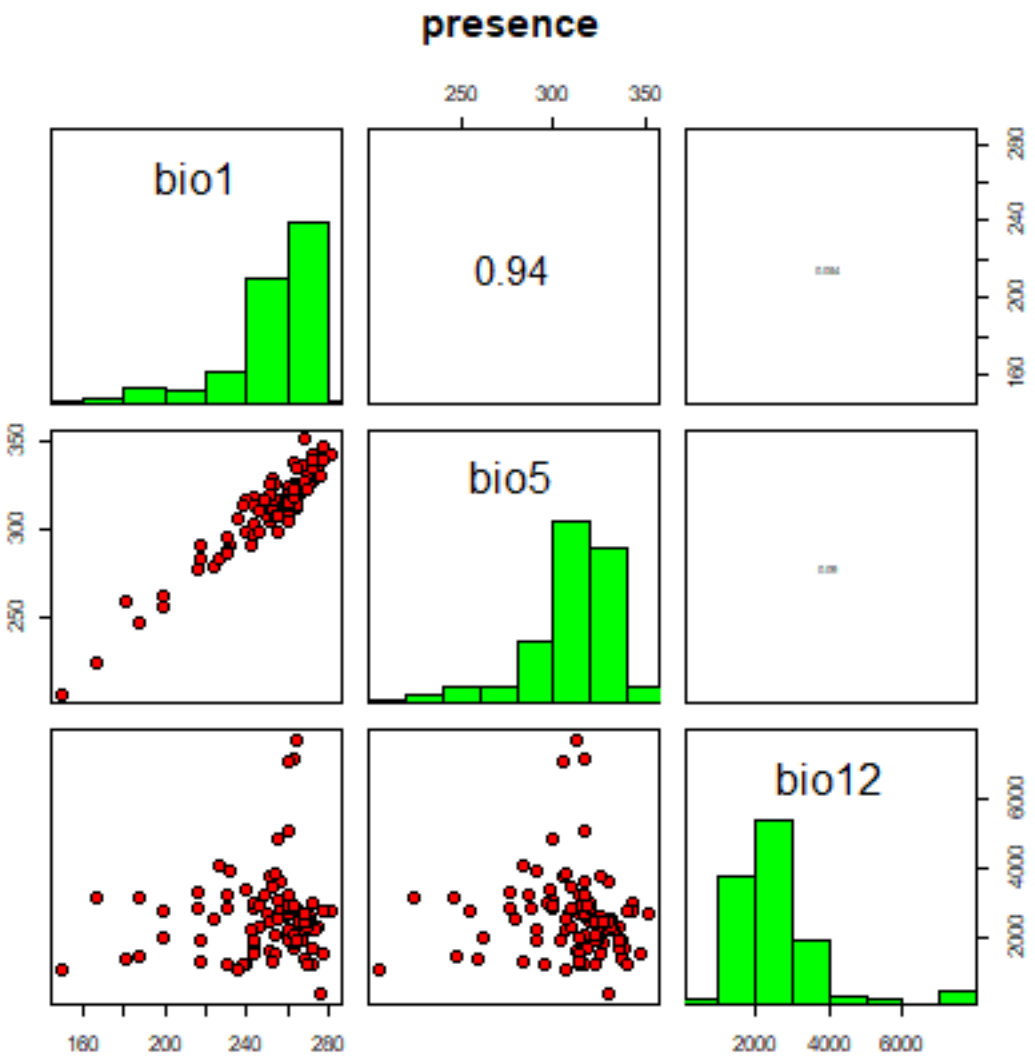
Models that are implemented in dismo do not use a formula (and most models only take presence points). Bioclim is an example. It only uses presence data, so we use 'presvals' instead of 'sdmdata'.

```
bc <- bioclim(presvals[,c('bio1', 'bio5', 'bio12')])
class(bc)
## [1] "Bioclim"
## attr(,"package")
## [1] "dismo"
bc
## class    : Bioclim
##
## variables: bio1 bio5 bio12
##
##
## presence points: 116
##    bio1 bio5 bio12
## 1   263  338  1639
## 2   263  338  1639
## 3   253  329  3624
## 4   243  318  1693
## 5   243  318  1693
## 6   252  326  2501
## 7   240  317  1214
## 8   275  335  2259
## 9   271  327  2212
## 10  274  329  2233
```

(continues on next page)

(continued from previous page)

```
##   (... ...  ...)
pairs(bc)
```



5.2 Model prediction

Different modeling methods return different type of 'model' objects (typically they have the same name as the modeling method used). All of these 'model' objects, irrespective of their exact class, can be used to with the `predict` function to make predictions for any combination of values of the independent variables. This is illustrated in the example below where we make predictions with the glm model object 'm1' and for bioclim model 'bc', for three records with values for variables bio1, bio5 and bio12 (the variables used in the example above to create the model objects).

```
bio1 = c(40, 150, 200)
bio5 = c(60, 115, 290)
```

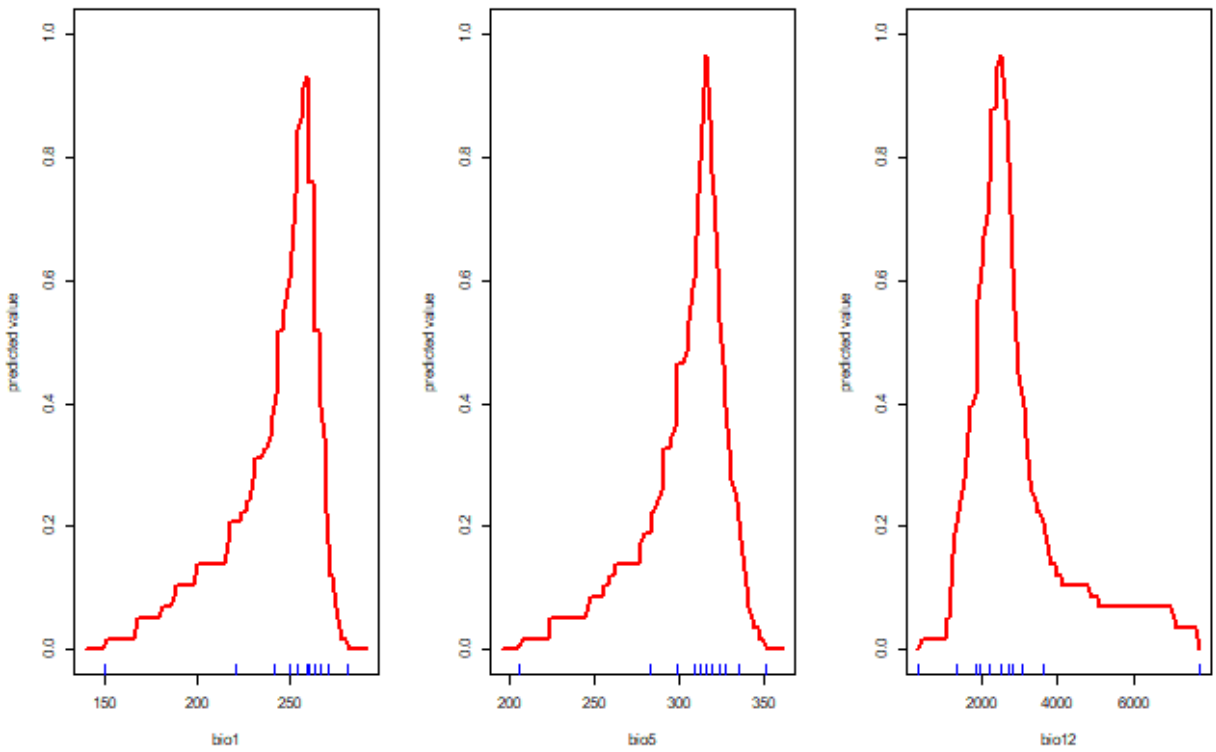
(continues on next page)

(continued from previous page)

```
bio12 = c(600, 1600, 1700)
pd = data.frame(cbind(bio1, bio5, bio12))
pd
##   bio1 bio5 bio12
## 1   40   60   600
## 2  150  115  1600
## 3  200  290  1700
predict(m1, pd)
##         1         2         3
## 0.1914136 0.3949569 0.2048905
predict(bc, pd)
```

Making such predictions for a few environments can be very useful to explore and understand model predictions. For example it used in the `response` function that creates response plots for each variable, with the other variables at their median value.

```
response(bc)
```



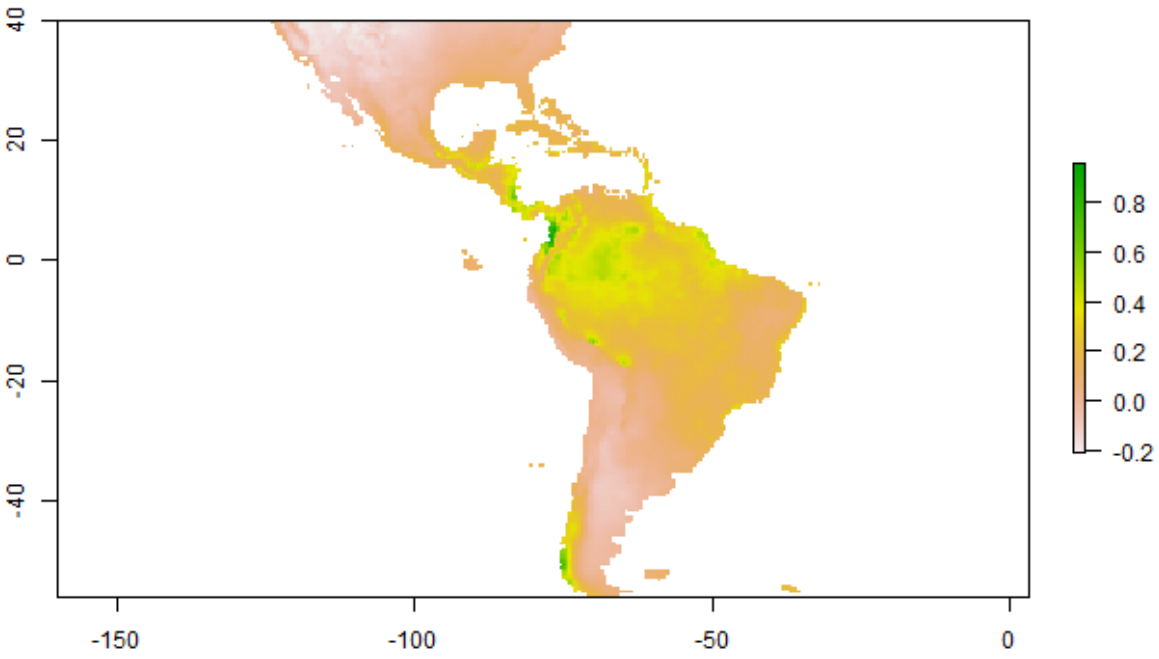
In most cases, howver, the purpose is SDM is to create a map of suitability scores. We can do that by providing the predict function with a Raster* object and a model object. As long as the variable names in the model object are available as layers (layerNames) in the Raster* object.

```
predictors <- stack(list.files(file.path(system.file(package="dismo"), 'ex'), pattern=
↪'grd$', full.names=TRUE ))
names(predictors)
## [1] "bio1"  "bio12" "bio16" "bio17" "bio5"  "bio6"  "bio7"  "bio8"  "biome"
```

(continues on next page)

(continued from previous page)

```
p <- predict(predictors, m1)
plot(p)
```



5.3 Model evaluation

It is much easier to create a model and make a prediction than to assess how good the model is, and whether it is can be used for a specific purpose. Most model types have different measures that can help to assess how good the model fits the data. It is worth becoming familiar with these and understanding their role, because they help you to assess whether there is anything substantially wrong with your model. Most statistics or machine learning texts will provide some details. For instance, for a GLM one can look at how much deviance is explained, whether there are patterns in the residuals, whether there are points with high leverage and so on. However, since many models are to be used for prediction, much evaluation is focused on how well the model predicts to points not used in model training (see following section on data partitioning). Before we start to give some examples of statistics used for this evaluation, it is worth considering what else can be done to evaluate a model. Useful questions include:

- Does the model seem sensible, ecologically?
- Do the fitted functions (the shapes of the modeled relationships) make sense?
- Do the predictions seem reasonable? (map them, and think about them)?
- Are there any spatial patterns in model residuals? (see Leathwick and Whitehead 2001 for an interesting example)

Most modelers rely on cross-validation. This consists of creating a model with one 'training' data set, and testing it with another data set of known occurrences. Typically, training and testing data are created through random sampling

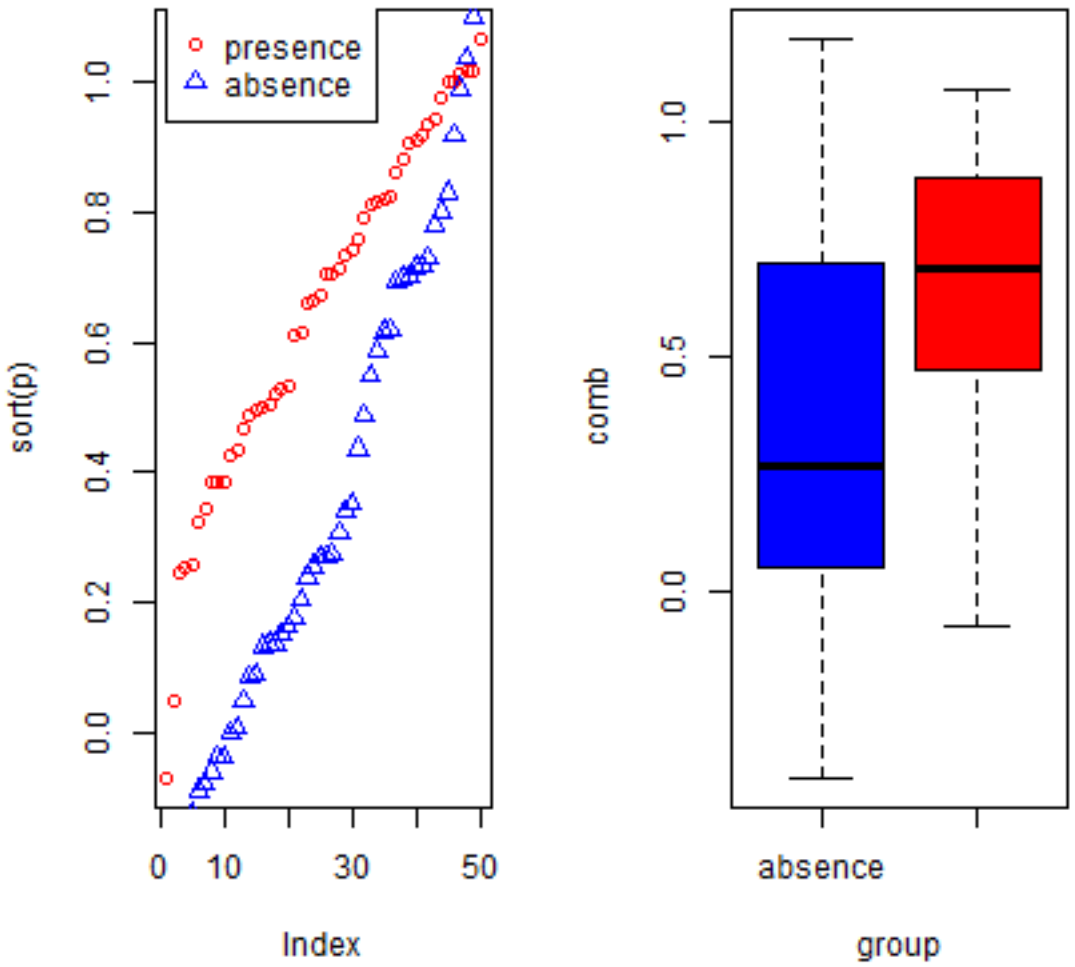
(without replacement) from a single data set. Only in a few cases, e.g. Elith *et al.*, 2006, training and test data are from different sources and pre-defined.

Different measures can be used to evaluate the quality of a prediction (Fielding and Bell, 1997, Liu et al., 2011; and Potts and Elith (2006) for abundance data), perhaps depending on the goal of the study. Many measures for evaluating models based on presence-absence or presence-only data are 'threshold dependent'. That means that a threshold must be set first (e.g., 0.5, though 0.5 is rarely a sensible choice – e.g. see Lui et al. 2005). Predicted values above that threshold indicate a prediction of 'presence', and values below the threshold indicate 'absence'. Some measures emphasize the weight of false absences; others give more weight to false presences.

Much used statistics that are threshold independent are the correlation coefficient and the Area Under the Receiver Operator Curve (AUROC, generally further abbreviated to AUC). AUC is a measure of rank-correlation. In unbiased data, a high AUC indicates that sites with high predicted suitability values tend to be areas of known presence and locations with lower model prediction values tend to be areas where the species is not known to be present (absent or a random point). An AUC score of 0.5 means that the model is as good as a random guess. See Phillips *et al.* (2006) for a discussion on the use of AUC in the context of presence-only rather than presence/absence data.

Here we illustrate the computation of the correlation coefficient and AUC with two random variables. `p` (presence) has higher values, and represents the predicted value for 50 known cases (locations) where the species is present, and `a` (absence) has lower values, and represents the predicted value for 50 known cases (locations) where the species is absent.

```
p <- rnorm(50, mean=0.7, sd=0.3)
a <- rnorm(50, mean=0.4, sd=0.4)
par(mfrow=c(1, 2))
plot(sort(p), col='red', pch=21)
points(sort(a), col='blue', pch=24)
legend(1, 0.95 * max(a,p), c('presence', 'absence'),
          pch=c(21,24), col=c('red', 'blue'))
comb <- c(p,a)
group <- c(rep('presence', length(p)), rep('absence', length(a)))
boxplot(comb~group, col=c('blue', 'red'))
```

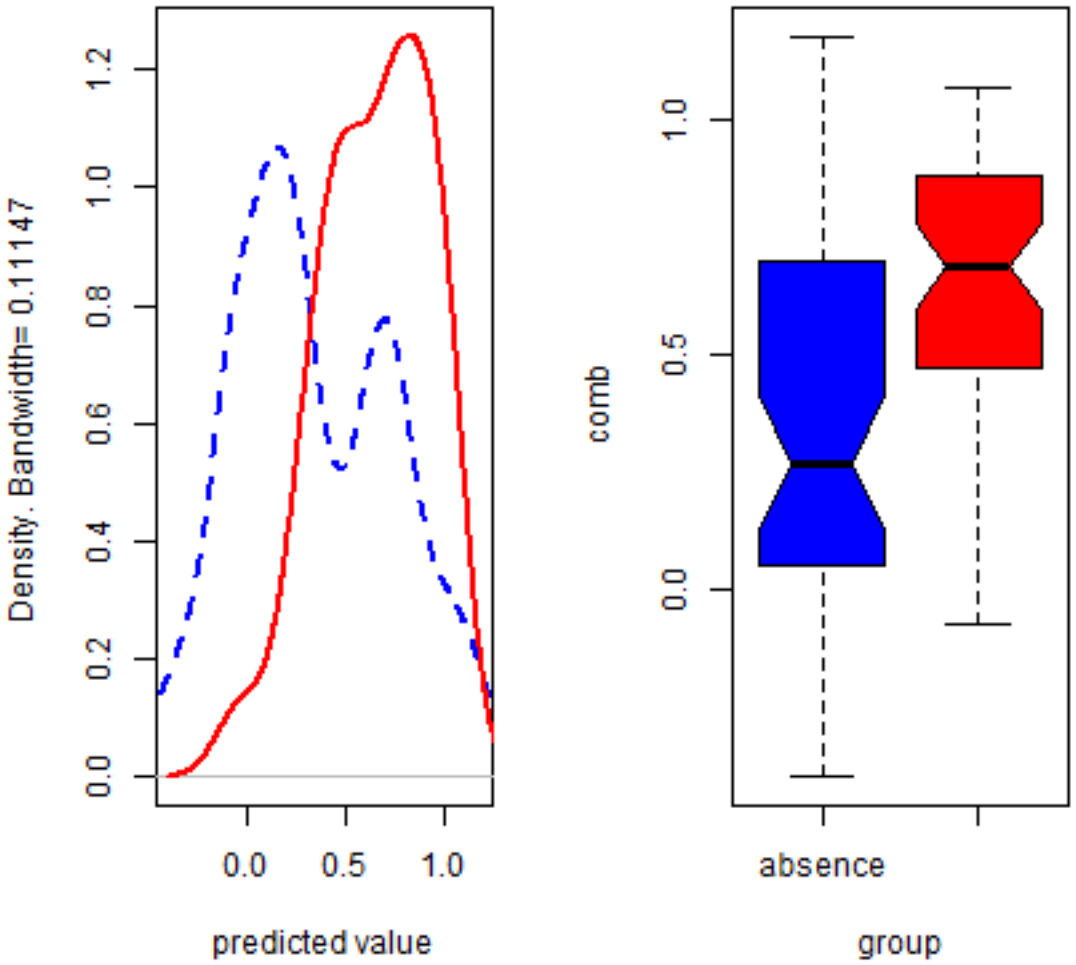


We created two variables with random normally distributed values, but with different mean and standard deviation. The two variables clearly have different distributions, and the values for 'presence' tend to be higher than for 'absence'. Here is how you can compute the correlation coefficient and the AUC:

```
group = c(rep(1, length(p)), rep(0, length(a)))
cor.test(comb, group)$estimate
##       cor
## 0.2594386
mv <- wilcox.test(p,a)
auc <- as.numeric(mv$statistic) / (length(p) * length(a))
auc
## [1] 0.6592
```

Below we show how you can compute these, and other statistics more conveniently, with the `evaluate` function in the dismo package. See `?evaluate` for info on additional evaluation measures that are available. ROC/AUC can also be computed with the `ROCR` package.

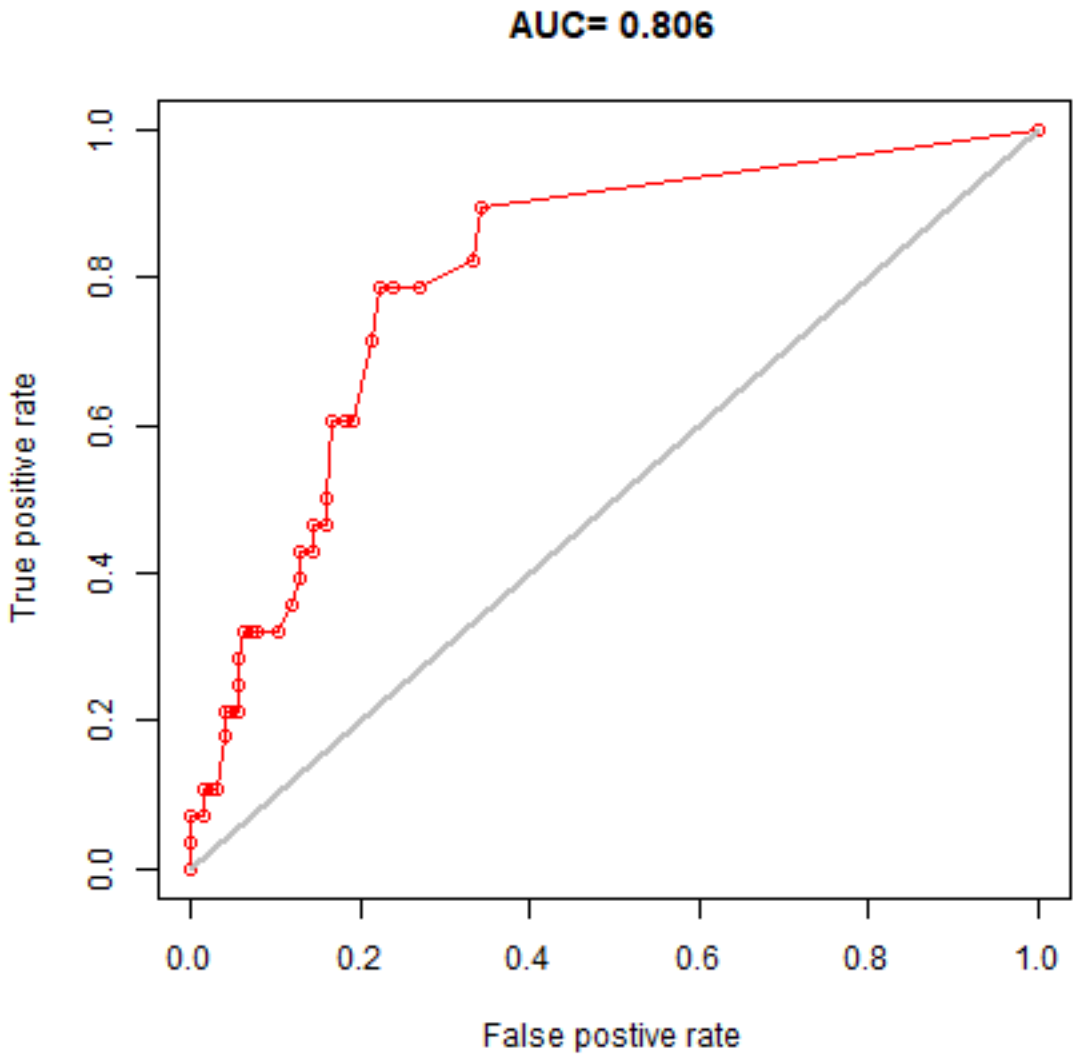
```
e <- evaluate(p=p, a=a)
class(e)
## [1] "ModelEvaluation"
## attr(,"package")
## [1] "dismo"
e
## class          : ModelEvaluation
## n presences    : 50
## n absences     : 50
## AUC            : 0.6592
## cor            : 0.2594386
## max TPR+TNR at : 0.4662509
par(mfrow=c(1, 2))
density(e)
boxplot(e, col=c('blue', 'red'))
```



Back to some real data, presence-only in this case. We'll divide the data in two random sets, one for training a Bioclim

model, and one for evaluating the model.

```
samp <- sample(nrow(sdmdata), round(0.75 * nrow(sdmdata)))
traindata <- sdmdata[samp,]
traindata <- traindata[traindata[,1] == 1, 2:9]
testdata <- sdmdata[-samp,]
bc <- bioclim(traindata)
e <- evaluate(testdata[testdata==1,], testdata[testdata==0,], bc)
e
## class          : ModelEvaluation
## n presences    : 24
## n absences     : 130
## AUC            : 0.8700321
## cor            : 0.4181033
## max TPR+TNR at : 0.0325087
plot(e, 'ROC')
```



In real projects, you would want to use k-fold data partitioning instead of a single random sample. The dismo function kfold facilitates that type of data partitioning. It creates a vector that assigns each row in the data matrix to a a group (between 1 to k).

Let's first create presence and background data.

```
pres <- sdmdata[sdmdata[,1] == 1, 2:9]
back <- sdmdata[sdmdata[,1] == 0, 2:9]
```

The background data will only be used for model testing and does not need to be partitioned. We now partition the data into 5 groups.

```
k <- 5
group <- kfold(pres, k)
group[1:10]
##  [1] 1 5 2 1 3 5 1 4 5 5
unique(group)
## [1] 1 5 2 3 4
```

Now we can fit and test our model five times. In each run, the records corresponding to one of the five groups is only used to evaluate the model, while the other four groups are only used to fit the model. The results are stored in a list called 'e'.

```
e <- list()
for (i in 1:k) {
    train <- pres[group != i,]
    test <- pres[group == i,]
    bc <- bioclim(train)
    e[[i]] <- evaluate(p=test, a=back, bc)
}
```

We can extract several things from the objects in 'e', but let's restrict ourselves to the AUC values and the "maximum of the sum of the sensitivity (true positive rate) and specificity (true negative rate)" threshold "spec_sens" (this is sometimes uses as a threshold for setting cells to presence or absence).

```
auc <- sapply(e, function(x){x@auc})
auc
## [1] 0.8321739 0.8374348 0.7620417 0.7329130 0.7622609
mean(auc)
## [1] 0.7853649
sapply( e, function(x){ threshold(x)['spec_sens'] } )
## $spec_sens
## [1] 0.07516882
##
## $spec_sens
## [1] 0.06441613
##
## $spec_sens
## [1] 0.06511739
##
## $spec_sens
## [1] 0.01065269
##
## $spec_sens
## [1] 0.03215806
```

The use of AUC in evaluating SDMs has been criticized (Lobo et al. 2008, Jiménez-Valverde 2011). A particularly sticky problem is that the values of AUC vary with the spatial extent used to select background points. Generally, the larger that extent, the higher the AUC value. Therefore, AUC values are generally biased and cannot be directly compared. Hijmans (2012) suggests that one could remove "spatial sorting bias" (the difference between the distance from testing-presence to training-presence and the distance from testing-absence to training-presence points) through "point-wise distance sampling".

```
file <- file.path(system.file(package="dismo"), "ex/bradypus.csv")
bradypus <- read.table(file,  header=TRUE,  sep=',')
bradypus <- bradypus[,-1]
presvals <- extract(predictors, bradypus)
set.seed(0)
backgr <- randomPoints(predictors, 500)

nr <- nrow(bradypus)
s <- sample(nr, 0.25 * nr)
pres_train <- bradypus[-s, ]
pres_test <- bradypus[s, ]

nr <- nrow(backgr)
set.seed(9)
s <- sample(nr, 0.25 * nr)
back_train <- backgr[-s, ]
back_test <- backgr[s, ]
```

```
sb <- ssb(pres_test, back_test, pres_train)
sb[,1] / sb[,2]
##         p
## 0.1000638
```

`sb[,1] / sb[,2]` is an indicator of spatial sorting bias (SSB). If there is no SSB this value should be 1, in these data it is close to zero, indicating that SSB is very strong. Let's create a subsample in which SSB is removed.

```
i <- pwdSample(pres_test, back_test, pres_train, n=1, tr=0.1)
pres_test_pwd <- pres_test[!is.na(i[,1]), ]
back_test_pwd <- back_test[na.omit(as.vector(i)), ]
sb2 <- ssb(pres_test_pwd, back_test_pwd, pres_train)
sb2[1]/ sb2[2]
## [1] 1.004106
```

Spatial sorting bias is much reduced now; notice how the AUC dropped!

```
bc <- bioclim(predictors, pres_train)
evaluate(bc, p=pres_test, a=back_test, x=predictors)
## class          : ModelEvaluation
## n presences    : 29
## n absences     : 125
## AUC            : 0.757931
## cor            : 0.2777298
## max TPR+TNR at : 0.03438276
evaluate(bc, p=pres_test_pwd, a=back_test_pwd, x=predictors)
## class          : ModelEvaluation
## n presences    : 19
## n absences     : 19
```

(continues on next page)

(continued from previous page)

```
## AUC             : 0.5914127
## cor             : 0.1487003
## max TPR+TNR at : 0.09185402
```

CHAPTER

SIX

MODELING METHODS

6.1 Types of algorithms and data used in examples

A large number of algorithms has been used in species distribution modeling. They can be classified as 'profile', 'regression', and 'machine learning' methods. Profile methods only consider 'presence' data, not absence or background data. Regression and machine learning methods use both presence and absence or background data. The distinction between regression and machine learning methods is not sharp, but it is perhaps still useful as way to classify models. Another distinction that one can make is between presence-only and presence-absence models. Profile methods are always presence-only, other methods can be either, depending if they are used with survey-absence or with pseudo-absence/background data. An entirely different class of models consists of models that only, or primarily, use the geographic location of known occurrences, and do not rely on the values of predictor variables at these locations. We refer to these models as 'geographic models'. Below we discuss examples of these different types of models.

First recreate the data we have used so far.

```
library(dismo)
## Loading required package: raster
library(maptools)
data(wrld_simpl)

predictors <- stack(list.files(file.path(system.file(package="dismo"), 'ex'), pattern=
↪'grd$', full.names=TRUE ))

file <- file.path(system.file(package="dismo"), "ex/bradypus.csv")
bradypus <- read.table(file,  header=TRUE,  sep=',')
bradypus <- bradypus[,-1]
presvals <- extract(predictors, bradypus)
set.seed(0)
backgr <- randomPoints(predictors, 500)
absvals <- extract(predictors, backgr)
pb <- c(rep(1, nrow(presvals)), rep(0, nrow(absvals)))
sdmdata <- data.frame(cbind(pb, rbind(presvals, absvals)))
sdmdata[,'biome'] <- as.factor(sdmdata[,'biome'])
```

We will use the same data to illustrate all models, except that some models cannot use categorical variables. So for those models we drop the categorical variables from the predictors stack.

```
pred_nf <- dropLayer(predictors, 'biome')
```

We use the `Bradypus` data for presence of a species. First we make a training and a testing set.

```
set.seed(0)
group <- kfold(bradypus, 5)
pres_train <- bradypus[group != 1, ]
pres_test <- bradypus[group == 1, ]
```

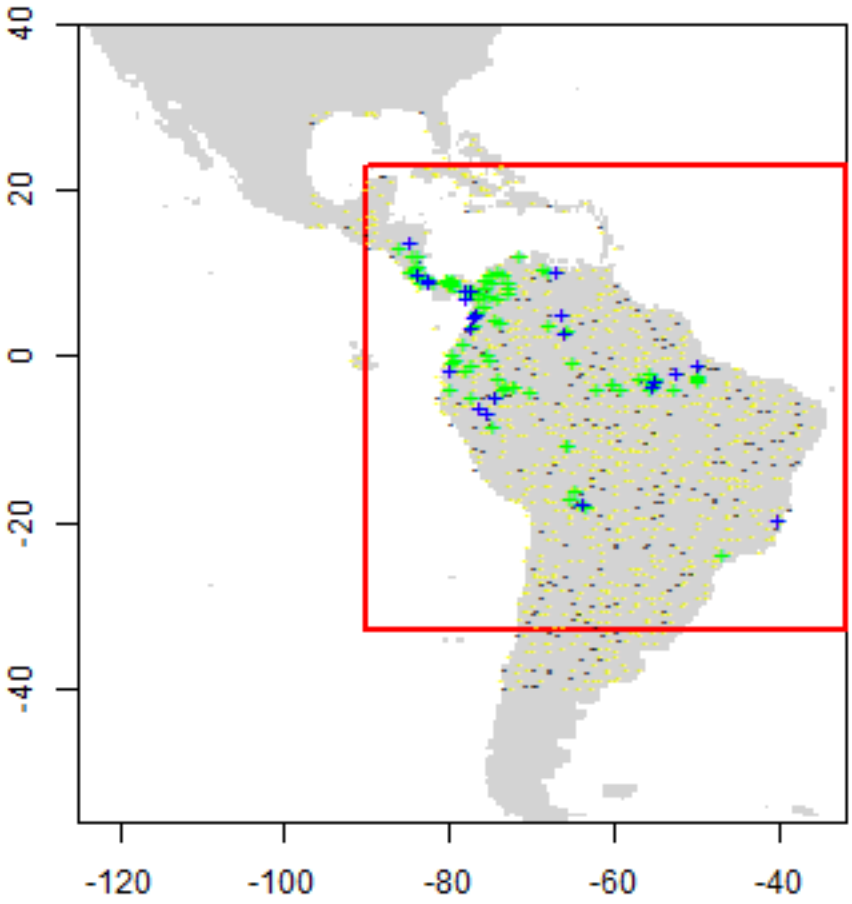
To speed up processing, let's restrict the predictions to a more restricted area (defined by a rectangular extent):

```
ext <- extent(-90, -32, -33, 23)
```

Background data for training and a testing set. The first layer in the RasterStack is used as a 'mask'. That ensures that random points only occur within the spatial extent of the rasters, and within cells that are not NA, and that there is only a single absence point per cell. Here we further restrict the background points to be within 12.5% of our specified extent 'ext'.

```
set.seed(10)
backg <- randomPoints(pred_nf, n=1000, ext=ext, extf = 1.25)
colnames(backg) = c('lon', 'lat')
group <- kfold(backg, 5)
backg_train <- backg[group != 1, ]
backg_test <- backg[group == 1, ]
```

```
r <- raster(pred_nf, 1)
plot(!is.na(r), col=c('white', 'light grey'), legend=FALSE)
plot(ext, add=TRUE, col='red', lwd=2)
points(backg_train, pch='-', cex=0.5, col='yellow')
points(backg_test, pch='-',  cex=0.5, col='black')
points(pres_train, pch= '+', col='green')
points(pres_test, pch='+', col='blue')
```



6.2 Profile methods

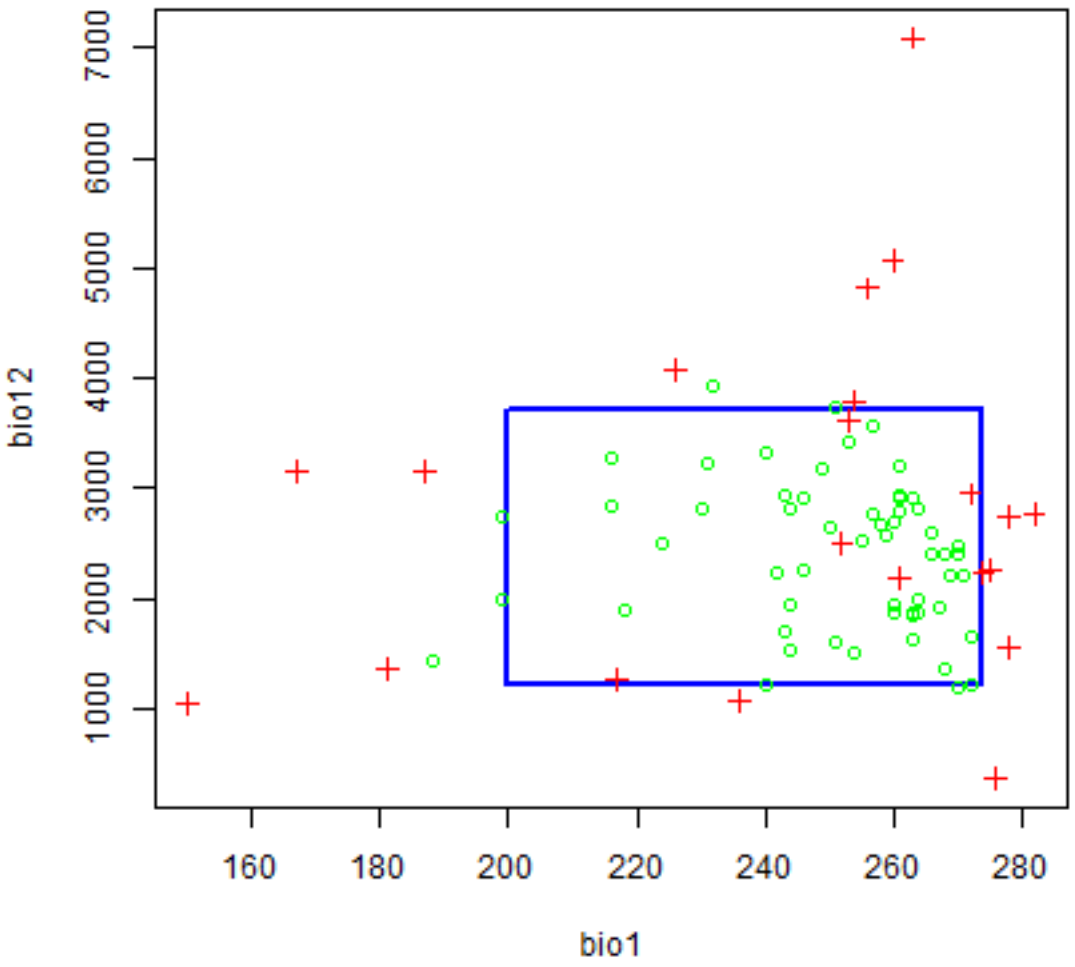
The three methods described here, Bioclim, Domain, and Mahal. These methods are implemented in the dismo package, and the procedures to use these models are the same for all three.

6.2.1 Bioclim

The BIOCLIM algorithm has been extensively used for species distribution modeling. BIOCLIM is a classic 'climate-envelope-model' (Booth *et al.*, 2014). Although it generally does not perform as good as some other modeling methods (Elith *et al.* 2006), particularly in the context of climate change (Hijmans and Graham, 2006), it is still used, among other reasons because the algorithm is easy to understand and thus useful in teaching species distribution modeling. The BIOCLIM algorithm computes the similarity of a location by comparing the values of environmental variables at any location to a percentile distribution of the values at known locations of occurrence ('training sites'). The closer to the 50th percentile (the median), the more suitable the location is. The tails of the distribution are not distinguished, that is, 10 percentile is treated as equivalent to 90 percentile. In the 'dismo' implementation, the values of the upper tail values are transformed to the lower tail, and the minimum percentile score across all the environmental variables is used (i.e., BIOCLIM uses an approach like Liebig's law of the minimum). This value is subtracted from 1 and then multiplied with two so that the results are between 0 and 1. The reason for scaling this way is that the results become more like that of other distribution modeling methods and are thus easier to interpret. The value 1 will rarely be observed as it would require a location that has the median value of the training data for all the variables considered. The value 0 is very common as it is assigned to all cells with a value of an environmental variable that is outside the percentile distribution (the range of the training data) for at least one of the variables.

Earlier on, we fitted a Bioclim model using data.frame with each row representing the environmental data at known sites of presence of a species. Here we fit a bioclim model simply using the predictors and the occurrence points (the function will do the extracting for us).

```
bc <- bioclim(pred_nf, pres_train)
plot(bc, a=1, b=2, p=0.85)
```



We evaluate the model in a similar way, by providing presence and background (absence) points, the model, and a RasterStack:

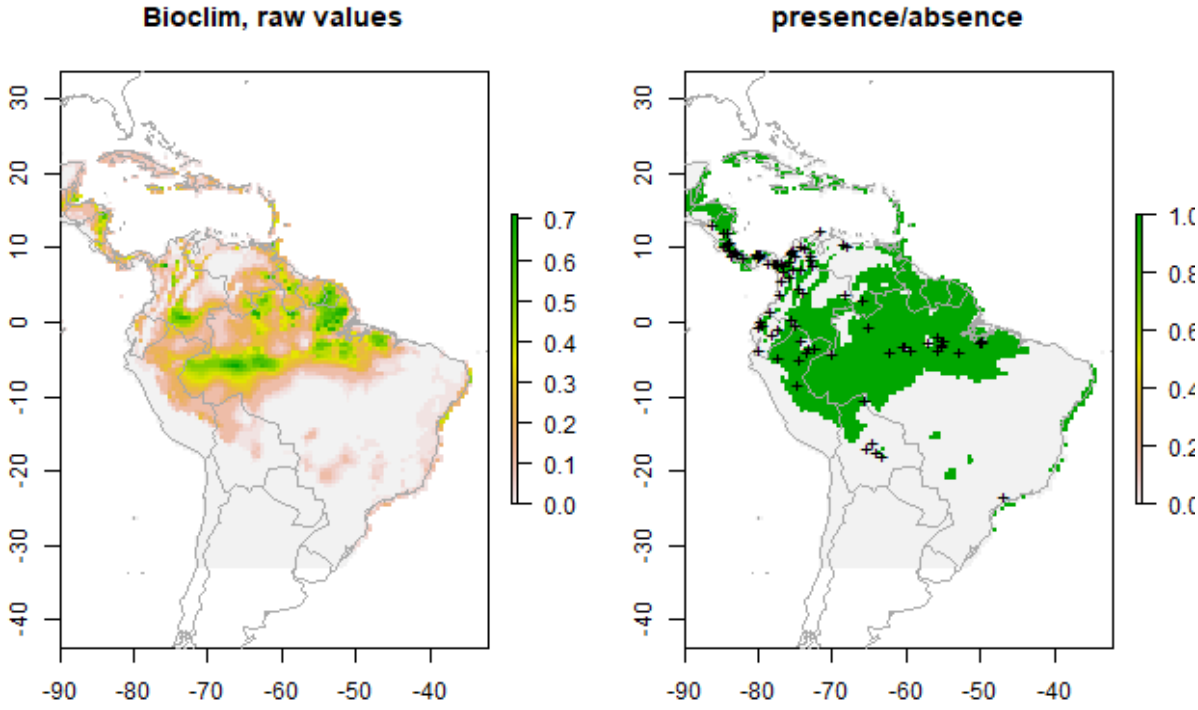
```
e <- evaluate(pres_test, backg_test, bc, pred_nf)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.6890217
## cor            : 0.1765706
## max TPR+TNR at : 0.08592151
```

Find a threshold

```
tr <- threshold(e, 'spec_sens')
tr
## [1] 0.08592151
```

And we use the RasterStack with predictor variables to make a prediction to a RasterLayer:

```
pb <- predict(pred_nf, bc, ext=ext, progress='')
pb
## class       : RasterLayer
## dimensions  : 112, 116, 12992  (nrow, ncol, ncell)
## resolution  : 0.5, 0.5  (x, y)
## extent      : -90, -32, -33, 23  (xmin, xmax, ymin, ymax)
## crs         : +proj=longlat +datum=WGS84 +no_defs
## source      : memory
## names       : layer
## values      : 0, 0.7096774  (min, max)
par(mfrow=c(1,2))
plot(pb, main='Bioclim, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
plot(pb > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```



Please note the order of the arguments in the predict function. In the example above, we used `predict(pred\_nf, bc)` (first the RasterStack, then the model object), which is little bit less efficient than predict(bc, pred_nf) (first the model, than the RasterStack). The reason for using the order we have used, is that this will work for all models, whereas the other option only works for the models defined in the dismo package, such as Bioclim, Domain, and Maxent, but not for models defined in other packages (random forest, boosted regression trees, glm, etc.).

6.2.2 Domain

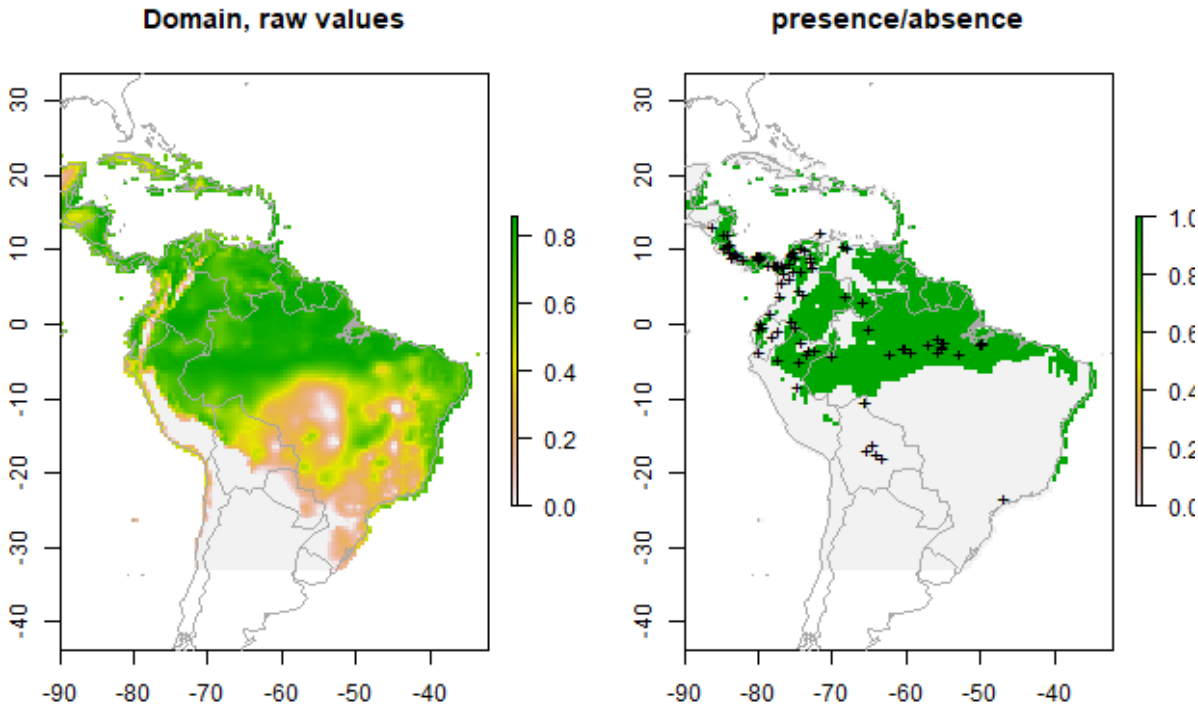
The Domain algorithm (Carpenter *et al.* 1993) has been extensively used for species distribution modeling. It did not perform very well in a model comparison (Elith *et al.* 2006) and very poorly when assessing climate change effects (Hijmans and Graham, 2006). The Domain algorithm computes the Gower distance between environmental variables at any location and those at any of the known locations of occurrence ('training sites').

The distance between the environment at point A and those of the known occurrences for a single climate variable is calculated as the absolute difference in the values of that variable divided by the range of the variable across all known occurrence points (i.e., the distance is scaled by the range of observations). For each variable the minimum distance between a site and any of the training points is taken. The Gower distance is then the mean of these distances over all environmental variables. The algorithm assigns to a place the distance to the closest known occurrence (in environmental space).

To integrate over environmental variables, the distance to any of the variables is used. This distance is subtracted from one, and (in this *R* implementation) values below zero are truncated so that the scores are between 0 (low) and 1 (high).

Below we fit a domain model, evaluate it, and make a prediction. We map the prediction, as well as a map subjectively classified into presence / absence.

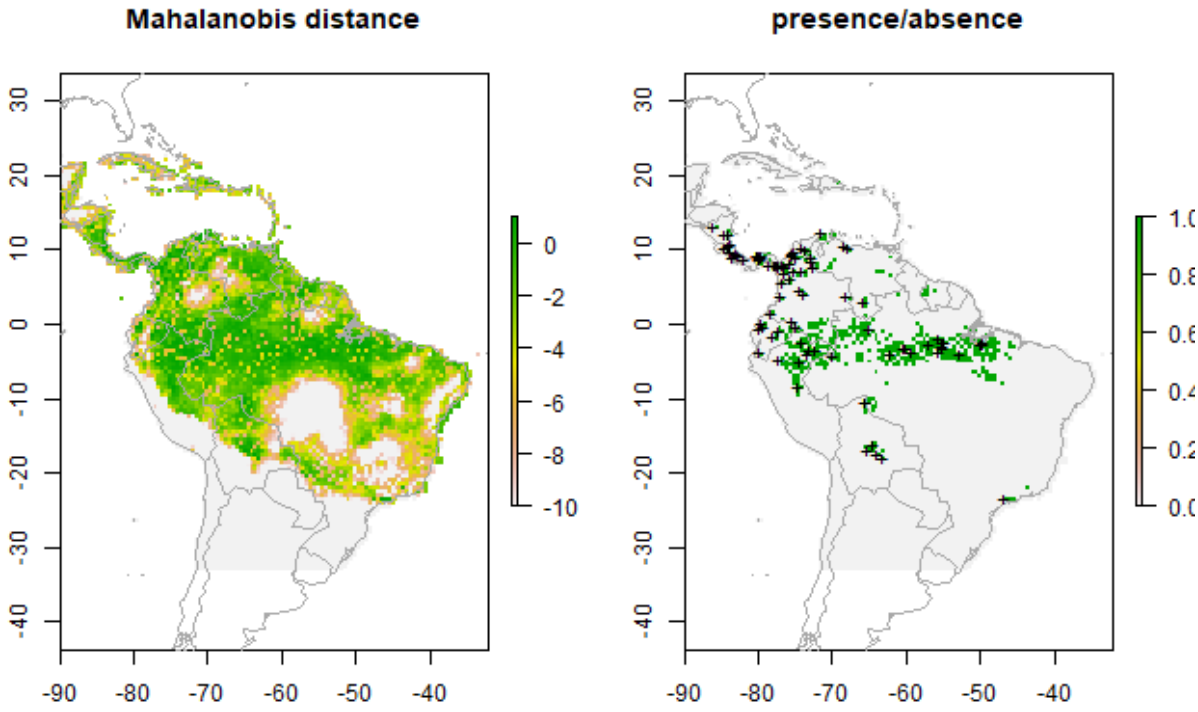
```
dm <- domain(pred_nf, pres_train)
e <- evaluate(pres_test, backg_test, dm, pred_nf)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.7097826
## cor            : 0.2138087
## max TPR+TNR at : 0.7107224
pd = predict(pred_nf, dm, ext=ext, progress='')
par(mfrow=c(1,2))
plot(pd, main='Domain, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(pd > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```



6.2.3 Mahalanobis distance

The `mahal` function implements a species distribution model based on the Mahalanobis distance (Mahalanobis, 1936). Mahalanobis distance takes into account the correlations of the variables in the data set, and it is not dependent on the scale of measurements.

```
mm <- mahal(pred_nf, pres_train)
e <- evaluate(pres_test, backg_test, mm, pred_nf)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.7686957
## cor            : 0.1506777
## max TPR+TNR at : 0.1116504
pm = predict(pred_nf, mm, ext=ext, progress='')
par(mfrow=c(1,2))
pm[pm < -10] <- -10
plot(pm, main='Mahalanobis distance')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(pm > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```



6.3 Classical regression models

The remaining models need to be fit with presence and absence (background) data. With the exception of 'maxent', we cannot fit the model with a RasterStack and points. Instead, we need to extract the environmental data values ourselves, and fit the models with these values.

```
train <- rbind(pres_train, backg_train)
pb_train <- c(rep(1, nrow(pres_train)), rep(0, nrow(backg_train)))
envtrain <- extract(predictors, train)
envtrain <- data.frame( cbind(pa=pb_train, envtrain) )
envtrain[,'biome'] = factor(envtrain[,'biome'], levels=1:14)
head(envtrain)
##   pa bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8 biome
## 1  1  263  1639   724    62  338  191  147  261     1
## 2  1  263  1639   724    62  338  191  147  261     1
## 3  1  253  3624  1547   373  329  150  179  271     1
## 4  1  243  1693   775   186  318  150  168  264     1
## 5  1  252  2501  1081   280  326  154  172  270     1
## 6  1  240  1214   516   146  317  150  168  261     2

testpres <- data.frame( extract(predictors, pres_test) )
testbackg <- data.frame( extract(predictors, backg_test) )
testpres[ ,'biome'] = factor(testpres[ ,'biome'], levels=1:14)
testbackg[ ,'biome'] = factor(testbackg[ ,'biome'], levels=1:14)
```

6.3.1 Generalized Linear Models

A generalized linear model (GLM) is a generalization of ordinary least squares regression. Models are fit using maximum likelihood and by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value. Depending on how a GLM is specified it can be equivalent to (multiple) linear regression, logistic regression or Poisson regression. See Guisan `et al` (2002) for an overview of the use of GLM in species distribution modeling.

In *R*, GLM is implemented in the 'glm' function, and the link function and error distribution are specified with the 'family' argument. Examples are:

```
family = binomial(link = "logit")
```

```
family = gaussian(link = "identity")
```

```
family = poisson(link = "log")
```

Here we fit two basic glm models. All variables are used, but without interaction terms.

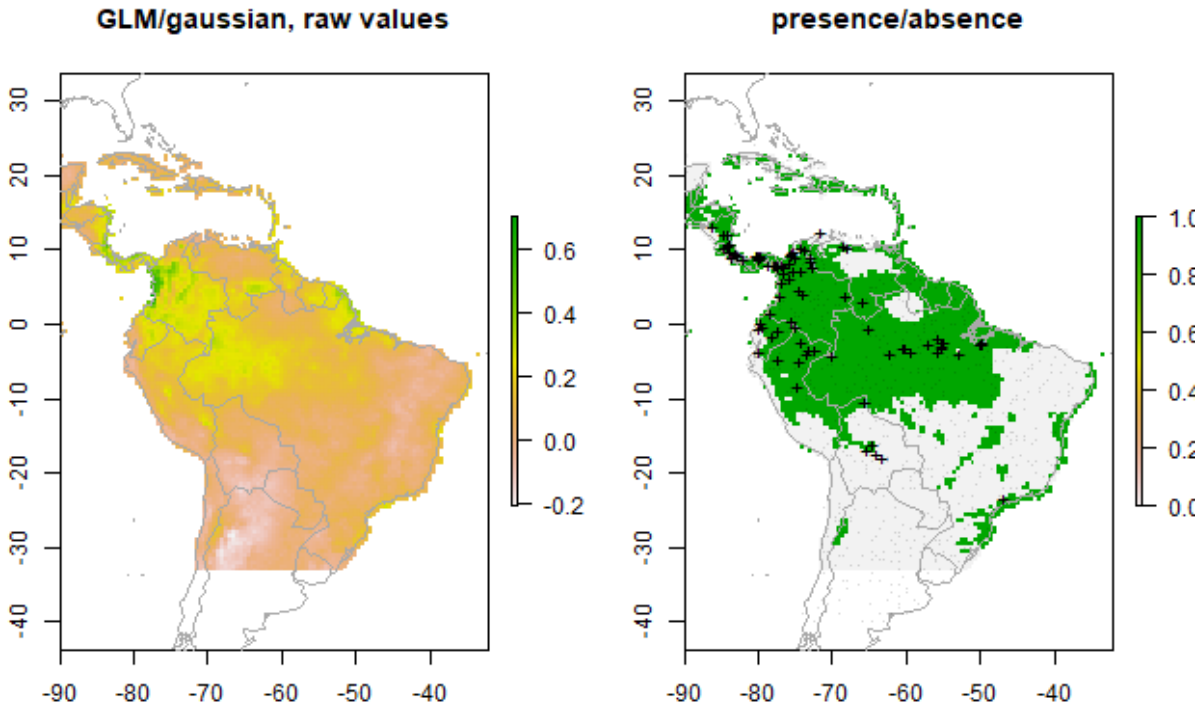
```
# logistic regression:
gm1 <- glm(pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17,
           family = binomial(link = "logit"), data=envtrain)

summary(gm1)
##
## Call:
## glm(formula = pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 +
##     bio16 + bio17, family = binomial(link = "logit"), data = envtrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73818  -0.49933  -0.23999  -0.06579   2.91501
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.0540581  1.5774812   1.936  0.05286 .
## bio1         0.0906378  0.0577068   1.571  0.11626
## bio5         0.2403921  0.2520843   0.954  0.34028
## bio6        -0.3227360  0.2541727  -1.270  0.20417
## bio7        -0.3212603  0.2528026  -1.271  0.20380
## bio8        -0.0133843  0.0255062  -0.525  0.59976
## bio12        0.0020951  0.0006931   3.023  0.00250 **
## bio16       -0.0023747  0.0014546  -1.633  0.10256
## bio17       -0.0047152  0.0015473  -3.047  0.00231 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 596.69  on 892  degrees of freedom
## Residual deviance: 449.77  on 884  degrees of freedom
## AIC: 467.77
##
## Number of Fisher Scoring iterations: 8
coef(gm1)
##  (Intercept)         bio1         bio5         bio6         bio7         bio8
```

(continues on next page)

(continued from previous page)

```
##  3.054058115  0.090637822  0.240392091 -0.322736029 -0.321260278 -0.013384258
##        bio12        bio16        bio17
##  0.002095136 -0.002374688 -0.004715157

gm2 <- glm(pa ~ bio1+bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17,
            family = gaussian(link = "identity"), data=envtrain)

evaluate(testpres, testbackg, gm1)
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.8156522
## cor            : 0.308183
## max TPR+TNR at : -2.565312
ge2 <- evaluate(testpres, testbackg, gm2)
ge2
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.7886957
## cor            : 0.3629842
## max TPR+TNR at : 0.08711959
```

```
pg <- predict(predictors, gm2, ext=ext)
par(mfrow=c(1,2))
plot(pg, main='GLM/gaussian, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(ge2, 'spec_sens')
plot(pg > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```



6.3.2 Generalized Additive Models

Generalized additive models (GAMs; Hastie and Tibshirani, 1990; Wood, 2006) are an extension to GLMs. In GAMs, the linear predictor is the sum of smoothing functions. This makes GAMs very flexible, and they can fit very complex functions. It also makes them very similar to machine learning methods. In *R*, GAMs are implemented in the 'mgcv' package.

6.4 Machine learning methods

Machine learning models, are non-parametric flexible regression models. Methods include Artifical Neural Networks (ANN), Random Forests, Boosted Regression Trees, and Support Vector Machines. Through the dismo package you can also use the Maxent program, that implements the most widely used method (maxent) in species distribution modeling. Breiman (2001a) provides a accessible introduction to machine learning, and how it contrasts with 'classical statistics' (model based probabilistic inference). Hastie *et al.*, 2009 provide what is probably the most extensive overview of these methods.

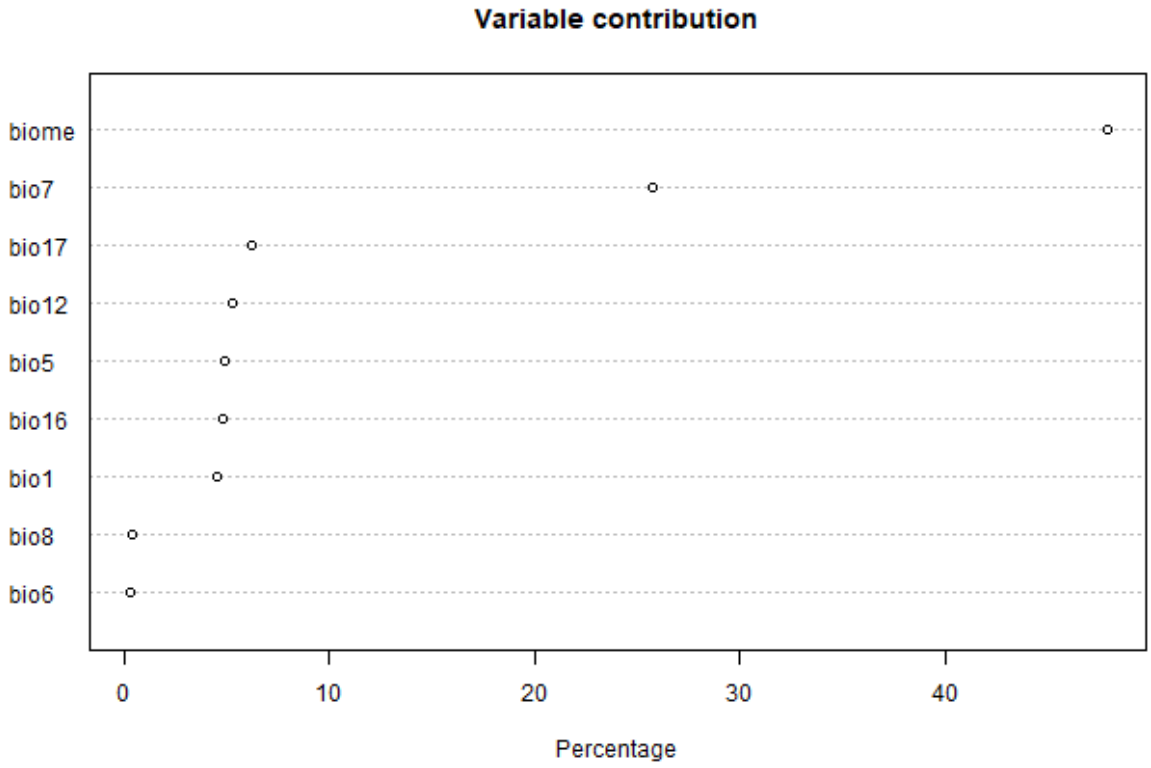
All the model fitting methods discussed here can be tuned in several ways. We do not explore that here, and only show the general approach. If you want to use one of the methods, then you should consult the *R* help pages (and other sources) to find out how to best implement the model fitting procedure.

6.4.1 Maxent

MaxEnt (short for "Maximum Entropy"; Phillips *et al.*, 2006) is the most widely used SDM algorithm. Elith *et al.* (2010) provide an explanation of the algorithm (and software) geared towards ecologists. MaxEnt is available as a stand-alone Java program. Dismo has a function 'maxent' that communicates with this program.

Because MaxEnt is implemented in dismo you can fit it like the profile methods (e.g. Bioclim). That is, you can provide presence points and a RasterStack. However, you can also first fit a model, like with the other methods such as glm. But in the case of MaxEnt you cannot use the formula notation.

```
maxent()
## Loading required namespace: rJava
## This is MaxEnt version 3.4.1
xm <- maxent(predictors, pres_train, factors='biome')
## This is MaxEnt version 3.4.1
plot(xm)
```



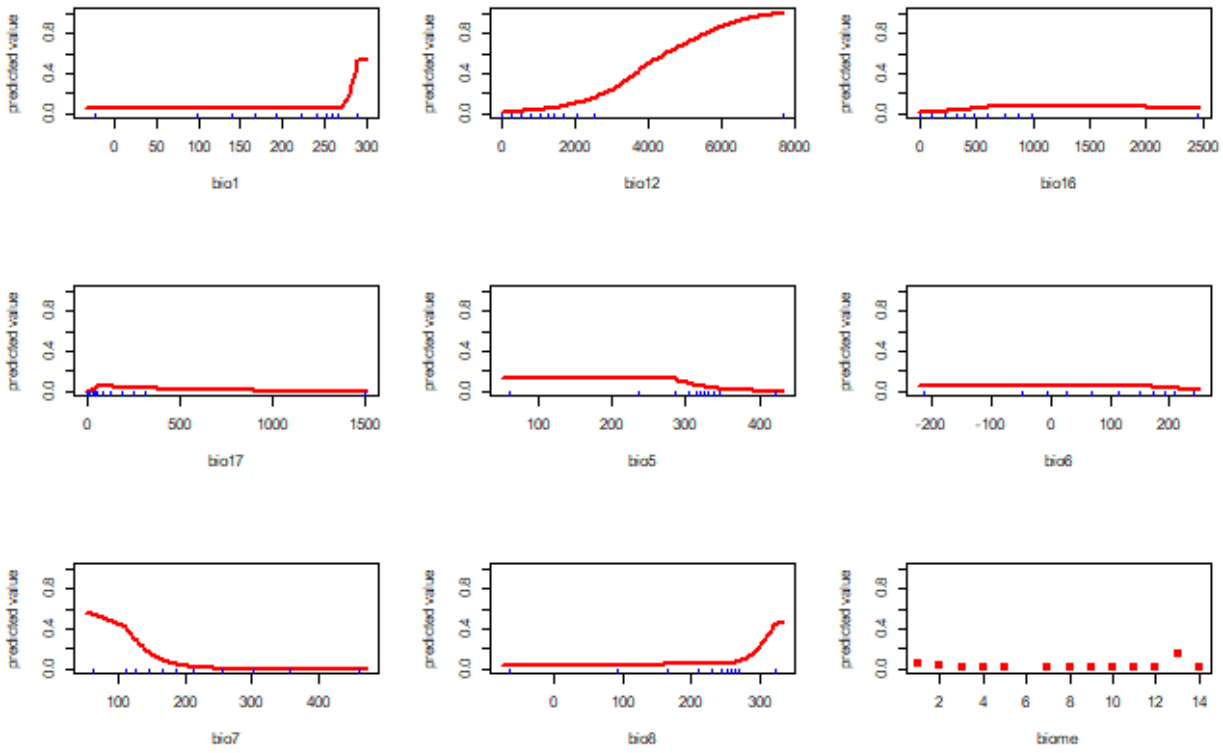
A response plot:

```
response(xm)
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
```

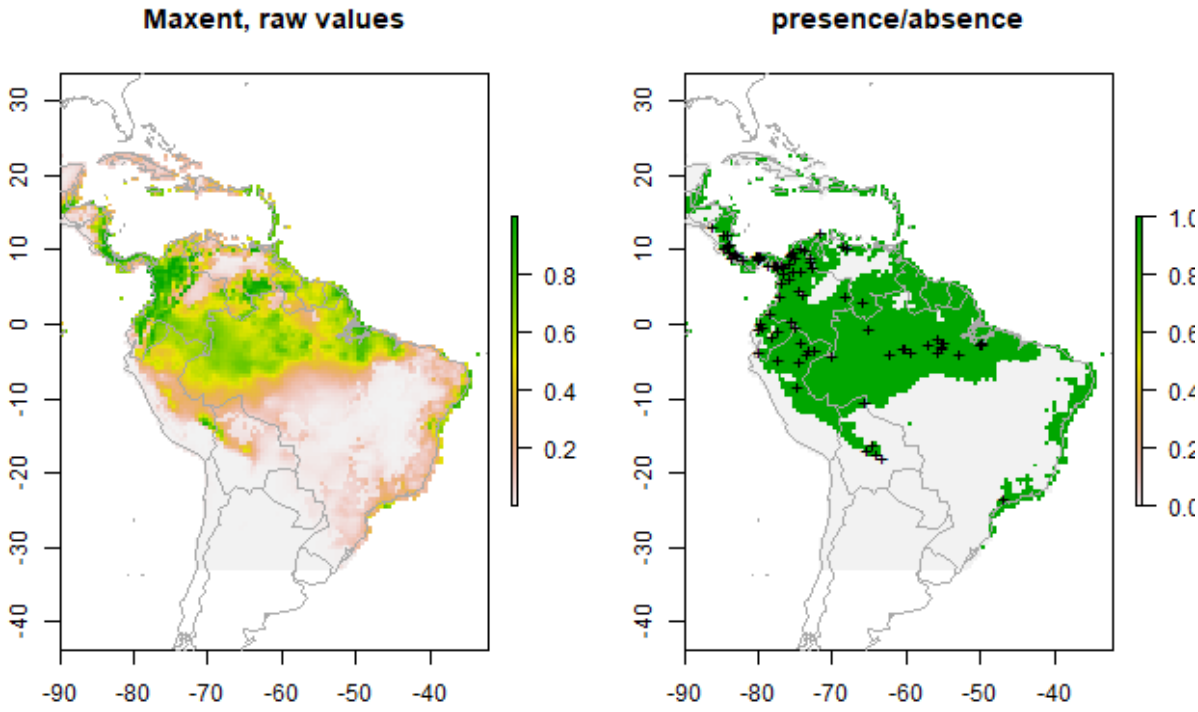
(continues on next page)

(continued from previous page)

```
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
```




```
e <- evaluate(pres_test, backg_test, xm, predictors)
## This is MaxEnt version 3.4.1
## This is MaxEnt version 3.4.1
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.8336957
## cor            : 0.3789954
## max TPR+TNR at : 0.1772358
px <- predict(predictors, xm, ext=ext, progress='')
## This is MaxEnt version 3.4.1
par(mfrow=c(1,2))
plot(px, main='Maxent, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(px > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```



6.4.2 Boosted Regression Trees

Boosted Regression Trees (BRT) is, unfortunately, known by a large number of different names. It was developed by Friedman (2001), who referred to it as a "Gradient Boosting Machine" (GBM). It is also known as "Gradient Boost", "Stochastic Gradient Boosting", "Gradient Tree Boosting". The method is implemented in the `gbm` package in *R*.

The article by Elith, Leathwick and Hastie (2009) describes the use of BRT in the context of species distribution modeling. Their article is accompanied by a number of R functions and a tutorial that have been slightly adjusted and incorporated into the 'dismo' package. These functions extend the functions in the `gbm` package, with the goal to make these easier to apply to ecological data, and to enhance interpretation. The adapted tutorial is available as an appendix.

6.4.3 Random Forest

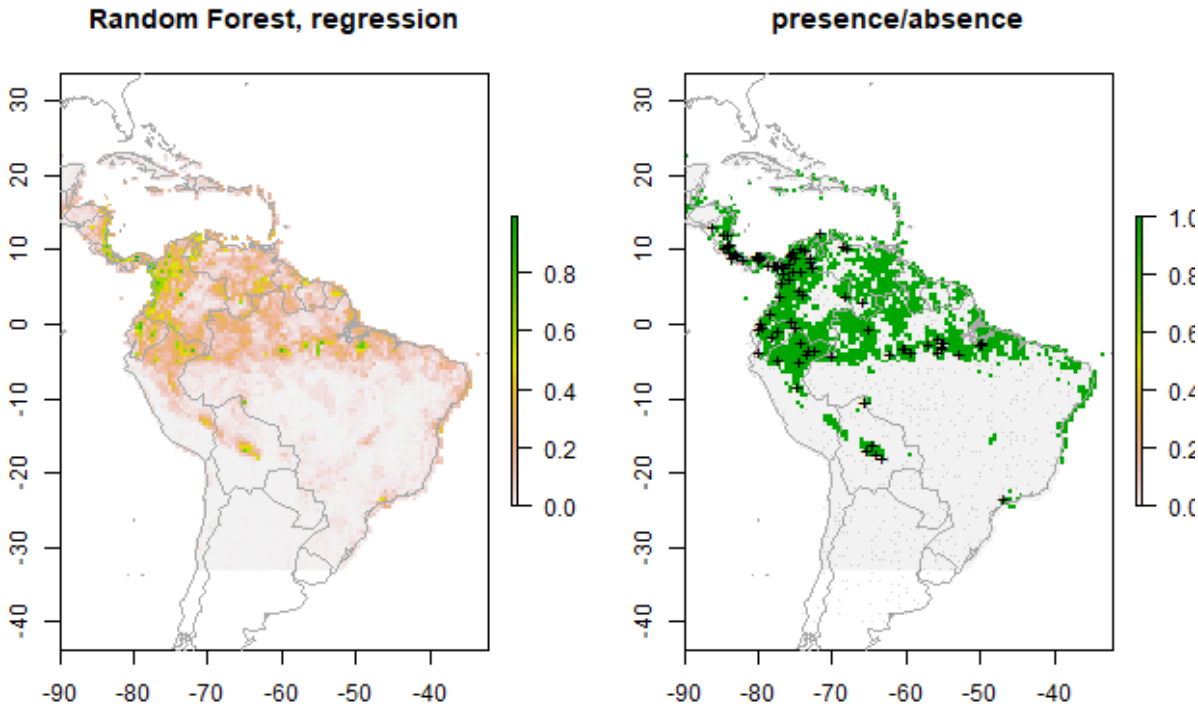
The Random Forest (Breiman, 2001b) method is an extension of Classification and regression trees (CART; Breiman *et al.*, 1984). In *R* it is implemented in the function 'randomForest' in a package with the same name. The function randomForest can take a formula or, in two separate arguments, a data.frame with the predictor variables, and a vector with the response. If the response variable is a factor (categorical), randomForest will do classification, otherwise it will do regression. Whereas with species distribution modeling we are often interested in classification (species is present or not), it is my experience that using regression provides better results. rf1 does regression, rf2 and rf3 do classification (they are exactly the same models). See the function tuneRF for optimizing the model fitting procedure.

```
library(randomForest)
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
model <- pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17
```

(continues on next page)

(continued from previous page)

```
rf1 <- randomForest(model, data=envtrain)
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
model <- factor(pa) ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17
rf2 <- randomForest(model, data=envtrain)
rf3 <- randomForest(envtrain[,1:8], factor(pb_train))

erf <- evaluate(testpres, testbackg, rf1)
erf
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.8580435
## cor            : 0.5010053
## max TPR+TNR at : 0.1060667

pr <- predict(predictors, rf1, ext=ext)

par(mfrow=c(1,2))
plot(pr, main='Random Forest, regression')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(erf, 'spec_sens')
plot(pr > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```



6.4.4 Support Vector Machines

Support Vector Machines (SVMs; Vapnik, 1998) apply a simple linear method to the data but in a high-dimensional feature space non-linearly related to the input space, but in practice, it does not involve any computations in that high-dimensional space. This simplicity combined with state of the art performance on many learning problems (classification, regression, and novelty detection) has contributed to the popularity of the SVM (Karatzoglou *et al.*, 2006). They were first used in species distribution modeling by Guo *et al.* (2005).

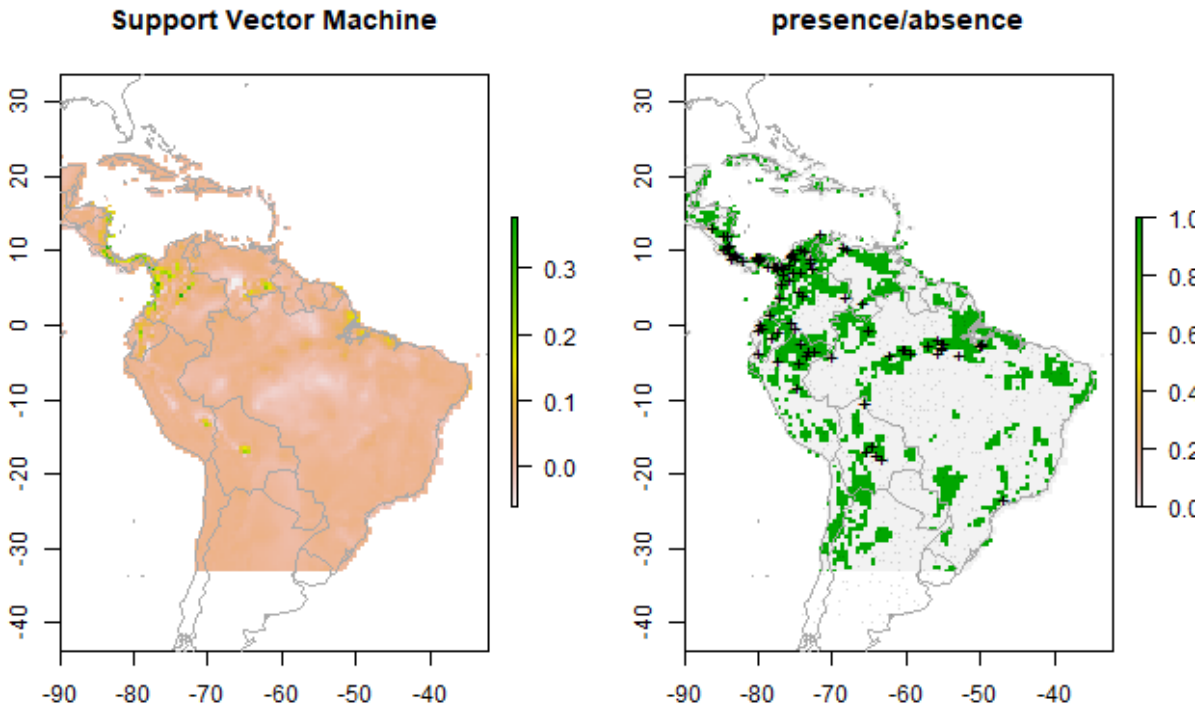
There are a number of implementations of svm in *R*. The most useful implementations in our context are probably function 'ksvm' in package 'kernlab' and the 'svm' function in package 'e1071'. 'ksvm' includes many different SVM formulations and kernels and provides useful options and features like a method for plotting, but it lacks a proper model selection tool. The 'svm' function in package 'e1071' includes a model selection tool: the 'tune' function (Karatzoglou *et al.*, 2006)

```
library(kernlab)
##
## Attaching package: 'kernlab'
## The following objects are masked from 'package:raster':
##
##     buffer, rotated
svm <- ksvm(pa ~ bio1+bio5+bio6+bio7+bio8+bio12+bio16+bio17, data=envtrain)
esv <- evaluate(testpres, testbackg, svm)
esv
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
```

(continues on next page)

(continued from previous page)

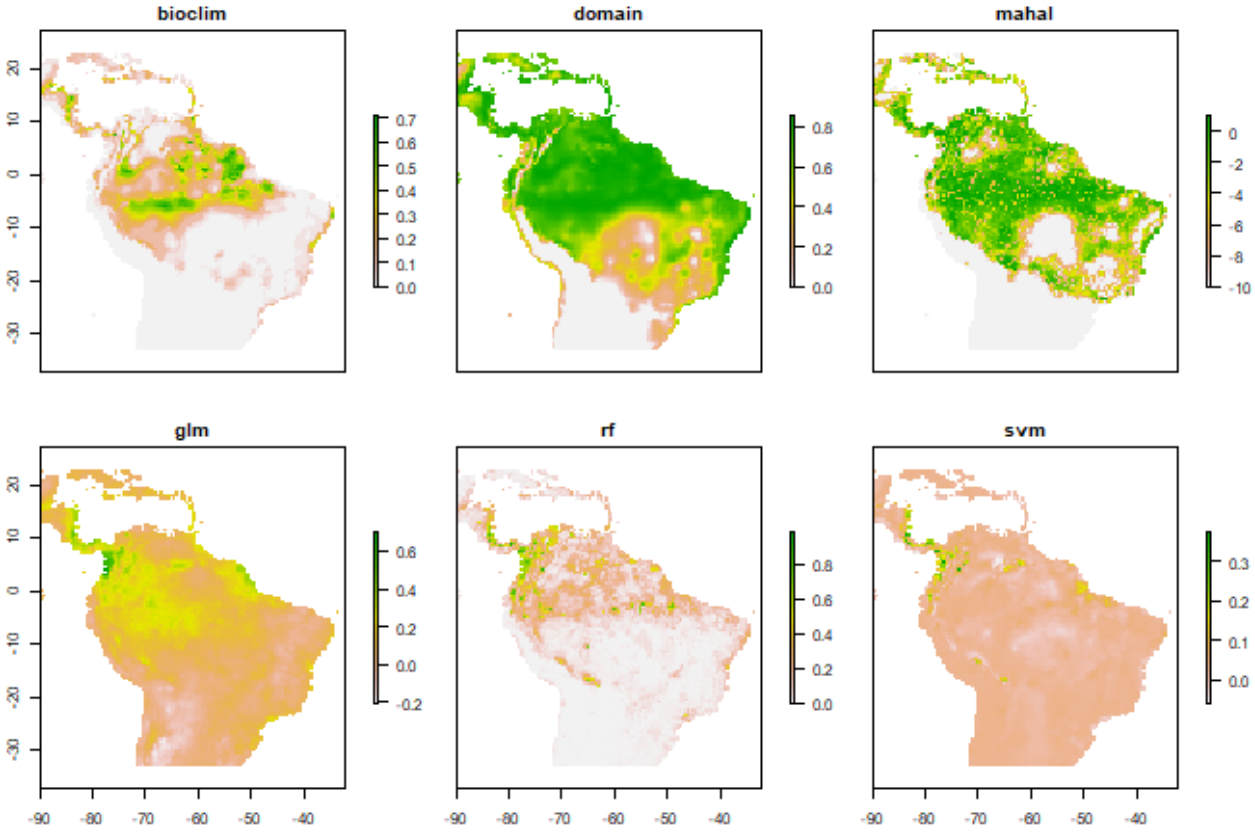
```
## AUC             : 0.7576087
## cor             : 0.3738667
## max TPR+TNR at : 0.02857293
ps <- predict(predictors, svm, ext=ext)

par(mfrow=c(1,2))
plot(ps, main='Support Vector Machine')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(esv, 'spec_sens')
plot(ps > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```



6.5 Combining model predictions

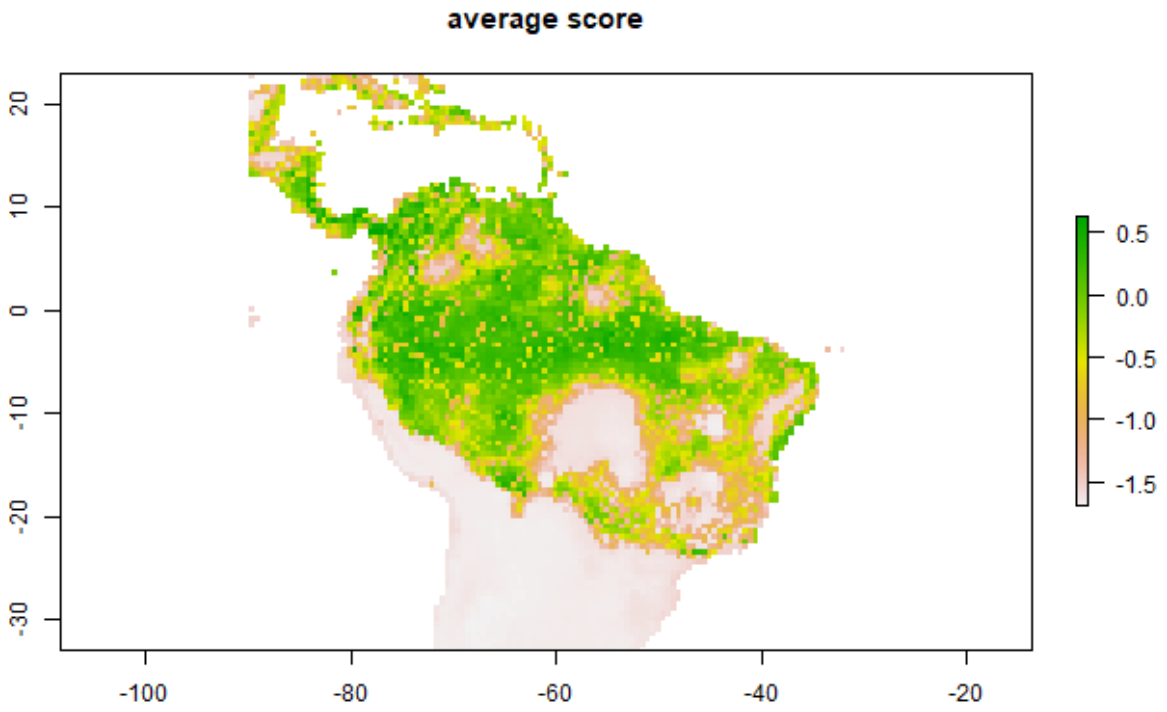
Rather than relying on a single "best" model, some auhtors (e.g. Thuillier, 2003) have argued for using many models and applying some sort of model averaging. See the biomod2 package for an implementation. You can of course implement these approaches yourself. Below is a very brief example. We first make a RasterStack of our individual model predictions:

```
models <- stack(pb, pd, pm, pg, pr, ps)
names(models) <- c("bioclim", "domain", "mahal", "glm", "rf", "svm")
plot(models)
```



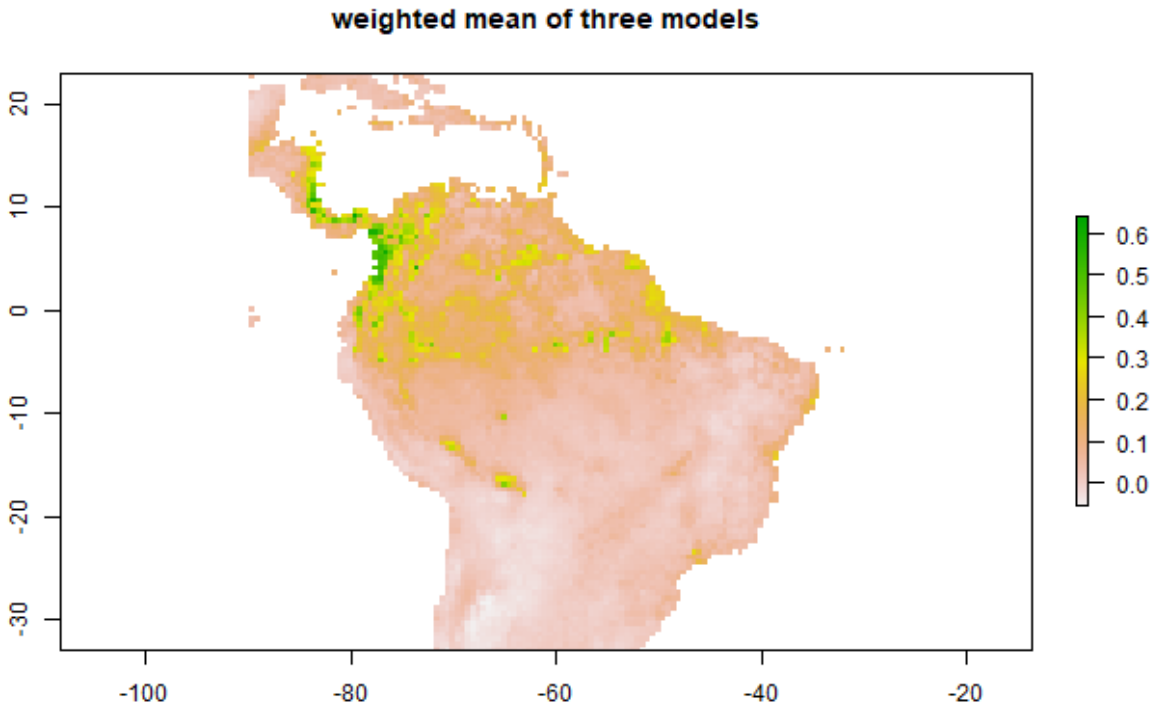
Now we can compute the simple average:

```
m <- mean(models)
plot(m, main='average score')
```



However, this is a problematic approach as the values predicted by the models are not all on the same (between 0 and 1) scale; so you may want to fix that first. Another concern could be weighting. Let's combine three models weighted by their AUC scores. Here, to create the weights, we substract 0.5 (the random expectation) and square the result to give further weight to higher AUC values.

```
auc <- sapply(list(ge2, erf, esv), function(x) x@auc)
w <- (auc-0.5)^2
m2 <- weighted.mean( models[[c("glm", "rf", "svm")]], w)
plot(m2, main='weighted mean of three models')
```


weighted mean of three models
20
10
0
-10
-20
-30
-100
-80
-60
-40
-20
0.6
0.5
0.4
0.3
0.2
0.1
0.0

CHAPTER

SEVEN

GEOGRAPHIC NULL MODELS

The 'geographic Null models' described here are not commonly used in species distribution modeling. They use the geographic location of known occurrences, and do not rely on the values of predictor variables at these locations. We are exploring their use in comparing and contrasting them with the other approaches (Bahn and McGill, 2007); in model evaluation as as null-models (Hijmans 2012); to sample background points; and generally to help think about the duality between geographic and environmental space (Collwel and Rangel, 2009). Below we show examples of these different types of models.

7.1 Geographic Distance

Simple model based on the assumption that the closer to a know presence point, the more likely it is to find the species.

Recreate our data.

```
library(dismo)
## Loading required package: raster
predictors <- stack(list.files(path=file.path(system.file(package="dismo"), 'ex'),␣
↪pattern='grd$', full.names=TRUE ))
ext <- extent(-90, -32, -33, 23)

bradypus <- read.csv(file.path(system.file(package="dismo"), "ex/bradypus.csv"))[,-1]
set.seed(0)
group <- kfold(bradypus, 5)
pres_train <- bradypus[group != 1, ]
pres_test <- bradypus[group == 1, ]

set.seed(0)
backgr <- randomPoints(predictors, 500)
set.seed(9)
nr <- nrow(backgr)
s <- sample(nr, 0.25 * nr)
back_train <- backgr[-s, ]
back_test <- backgr[s, ]

set.seed(10)
pred_nf <- dropLayer(predictors, 'biome')
backg <- randomPoints(pred_nf, n=1000, ext=ext, extf = 1.25)
colnames(backg) = c('lon', 'lat')
group <- kfold(backg, 5)
backg_train <- backg[group != 1, ]
```

(continues on next page)

(continued from previous page)

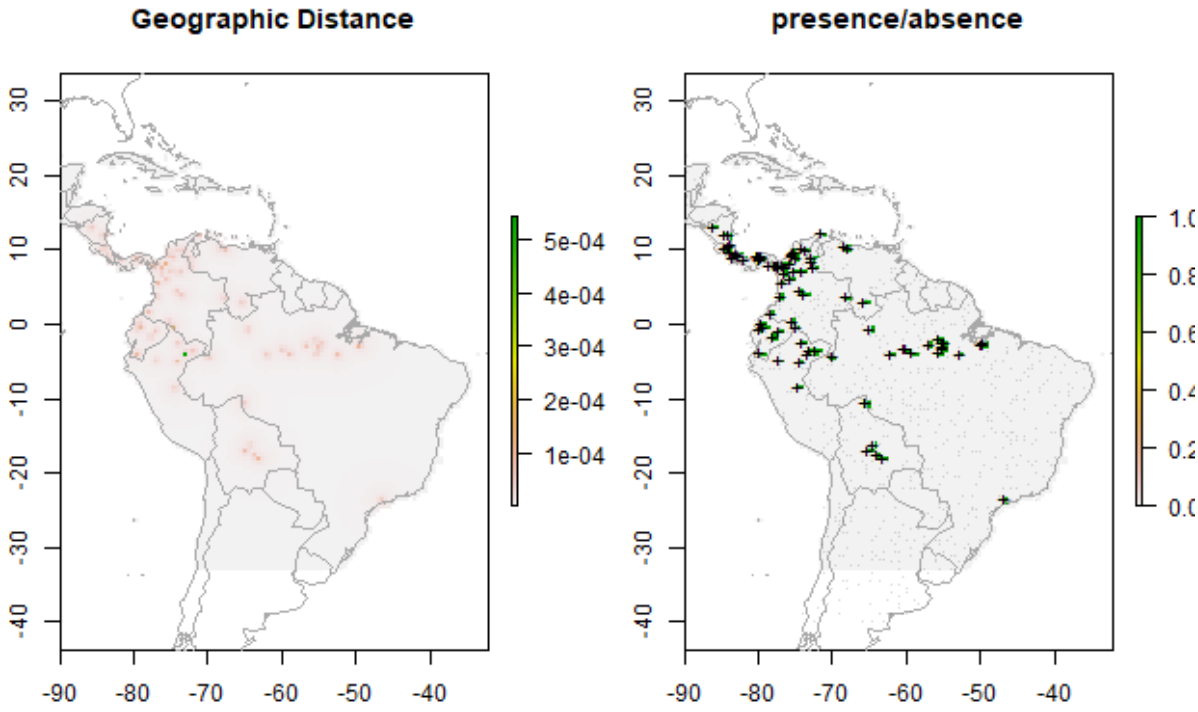
```
backg_test <- backg[group == 1, ]

library(maptools)
data(wrld_simpl)
```

First create a mask to predict to, and to use as a mask to only predict to land areas.

```
seamask <- crop(predictors[[1]], ext)
distm <- geoDist(pres_train, lonlat=TRUE)
ds <- predict(seamask, distm, mask=TRUE)
e <- evaluate(distm, p=pres_test, a=backg_test)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.9015217
## cor            : 0.4564334
## max TPR+TNR at : 2.217e-05
```

And the plots.

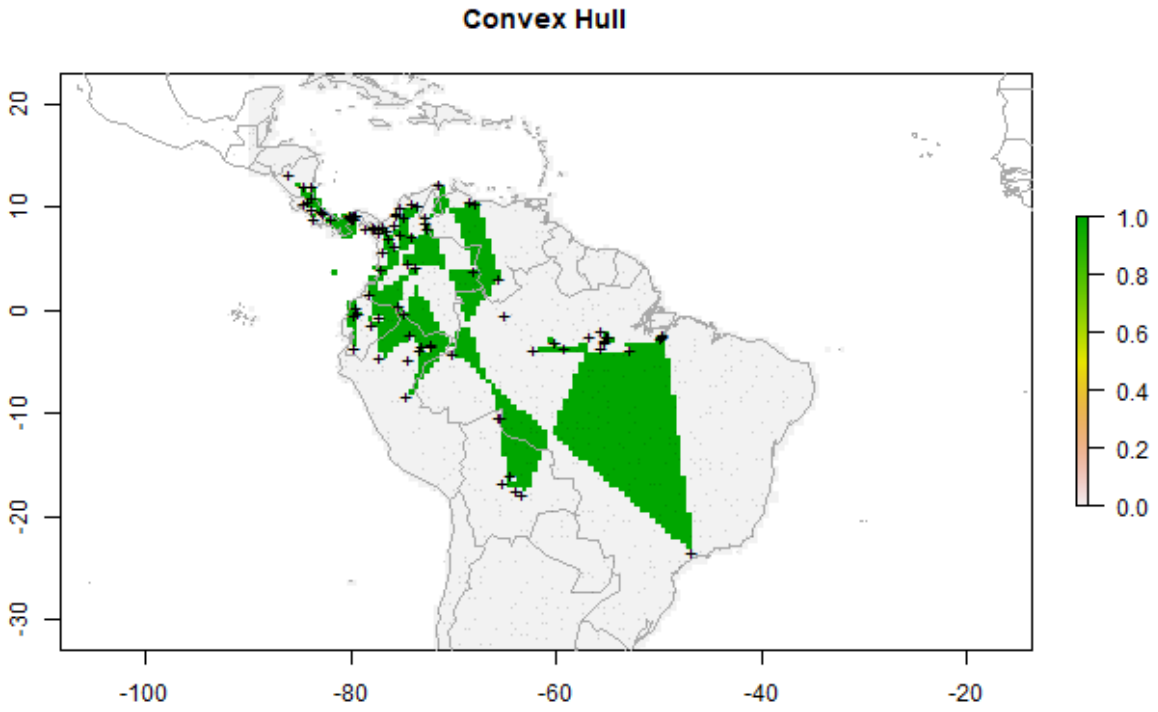
```
par(mfrow=c(1,2))
plot(ds, main='Geographic Distance')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(ds > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```



7.2 Convex hulls

This model draws a convex hull around all 'presence' points.

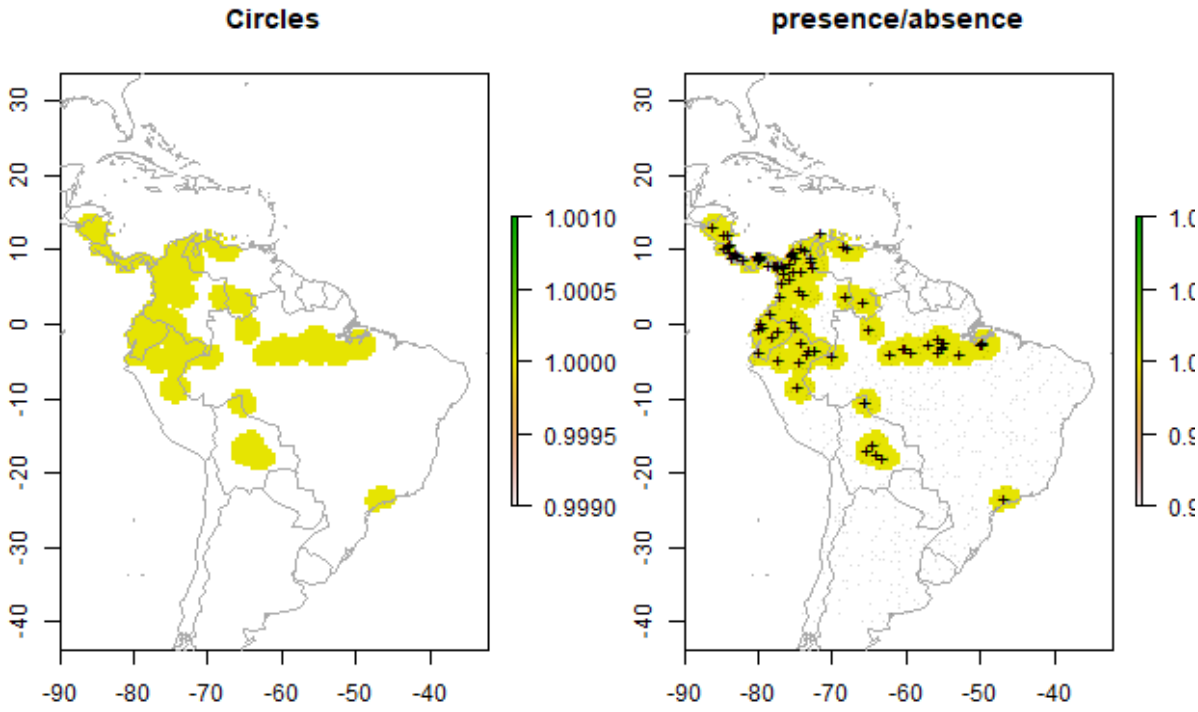
```
hull <- convHull(pres_train, lonlat=TRUE)
e <- evaluate(hull, p=pres_test, a=backg_test)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.5646739
## cor            : 0.1006176
## max TPR+TNR at : 0.9999

h <- predict(seamask, hull, mask=TRUE)

plot(h, main='Convex Hull')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```



7.3 Circles

This model draws circles around all 'presence' points.

```
circ <- circles(pres_train, lonlat=TRUE)
## Loading required namespace: rgeos
pc <- predict(seamask, circ, mask=TRUE)

e <- evaluate(circ, p=pres_test, a=backg_test)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.8297826
## cor            : 0.447742
## max TPR+TNR at : 0.9999

par(mfrow=c(1,2))
plot(pc, main='Circles')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(pc > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```



7.4 Presence/absence

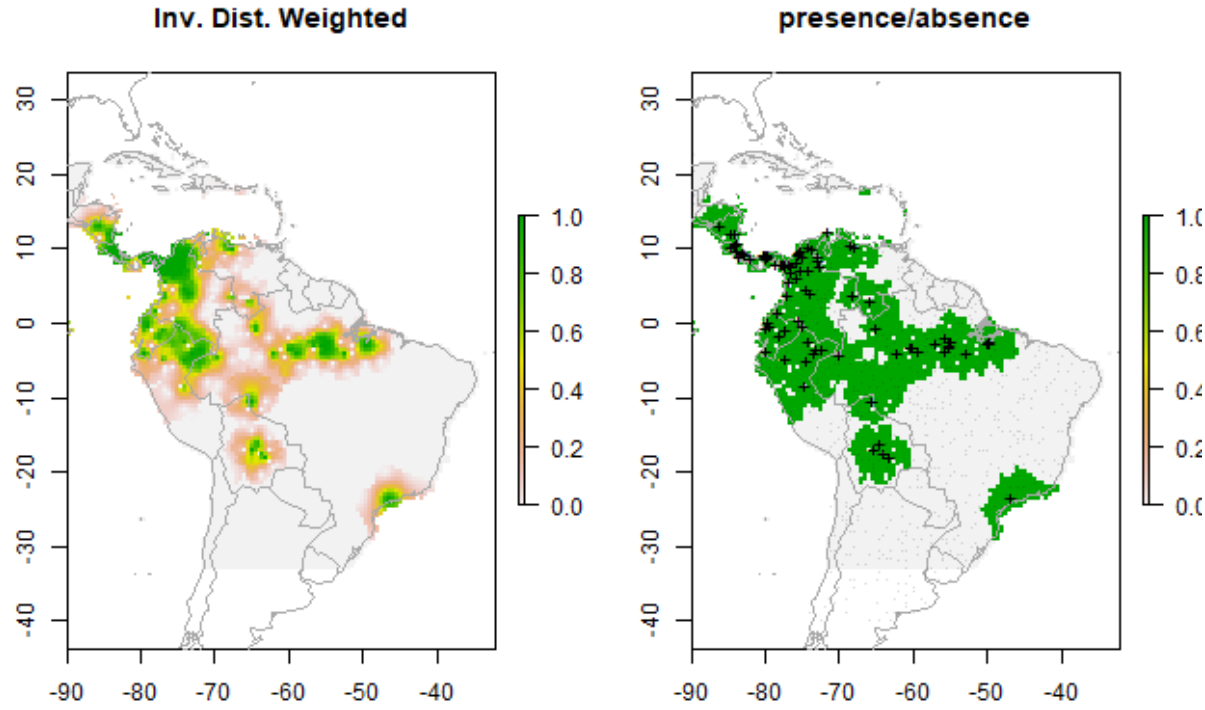
Spatial-only models for presence/background (or absence) data are also available through functions `geoIDW`, `voronoiHull`, and general geostatistical methods such as indicator kriging (available in the "gstat" package).

```
idwm <- geoIDW(p=pres_train, a=data.frame(back_train))
## Loading required namespace: gstat

e <- evaluate(idwm, p=pres_test, a=backg_test)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.8702174
## cor            : 0.4548002
## max TPR+TNR at : 0.03483593

iw <- predict(seamask, idwm, mask=TRUE)

par(mfrow=c(1,2))

plot(iw, main='Inv. Dist. Weighted')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
```

(continues on next page)

(continued from previous page)

```
pa <- mask(iw > tr, seamask)
plot(pa, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```

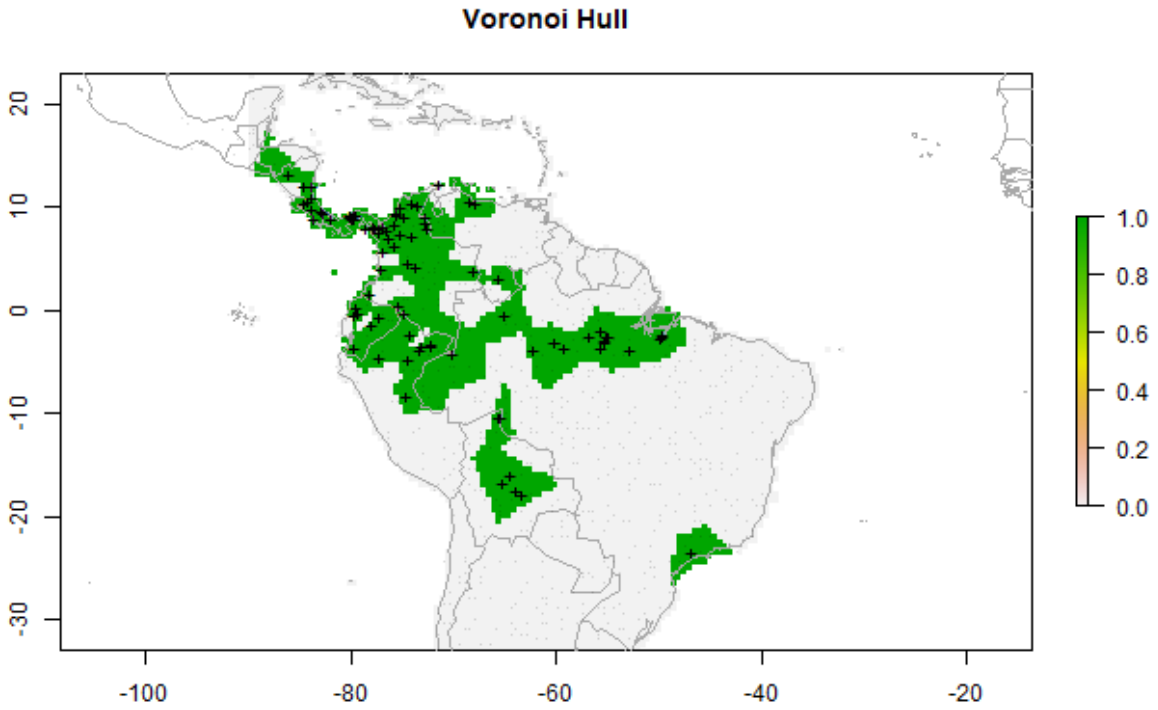



```
# take a smallish sample of the background training data
va <- data.frame(back_train[sample(nrow(back_train), 100), ])
vorm <- voronoiHull(p=pres_train, a=va)
## Loading required namespace: deldir

e <- evaluate(vorm, p=pres_test, a=backg_test)
e
## class          : ModelEvaluation
## n presences    : 23
## n absences     : 200
## AUC            : 0.49
## cor            : -0.04326367
## max TPR+TNR at : 0.9999

vo <- predict(seamask, vorm, mask=T)

plot(vo, main='Voronoi Hull')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```

(continues on next page)

(continued from previous page)

```
points(backg_train, pch='-', cex=0.25)
```



CHAPTER

EIGHT

REFERENCES

- Austin M.P., 2002. Spatial prediction of species distribution: an interface between ecological theory and statistical modelling. Ecological Modelling 157: 101-18.
- Austin, M.P., and T.M. Smith, 1989. A new model for the continuum concept. Vegetatio 83: 35-47.
- Bahn, V., and B.J. McGill, 2007. Can niche-based distribution models outperform spatial interpolation? Global Ecology and Biogeography 16: 733-742.
- Booth, T.H., H.A. Nix, J.R. Busby and M.F. Hutchinson, 2014. BIOCLIM: the first species distribution modelling package, its early applications and relevance to most current MAXENT studies. Diversity and Distributions 20: 1-9.
- Breiman, L., 2001a. Statistical Modeling: The Two Cultures. Statistical Science 16: 199-215.
- Breiman, L., 2001b. Random Forests. Machine Learning 45: 5-32.
- Breiman, L., J. Friedman, C.J. Stone and R.A. Olshen, 1984. Classification and Regression Trees. Chapman & Hall/CRC.
- Carpenter G., A.N. Gillison and J. Winter, 1993. Domain: a flexible modelling procedure for mapping potential distributions of plants and animals. Biodiversity Conservation 2: 667-680.
- Colwell R.K. and T.F. Rangel, 2009. Hutchinson's duality: The once and future niche. Proceedings of the National Academy of Sciences 106: 19651-19658.
- Dormann C.F., Elith J., Bacher S., Buchmann C., Carl G., Carré G., Diekötter T., García Marquéz J., Gruber B., Lafourcade B., Leitão P.J., Münkemüller T., McClean C., Osborne P., Reineking B., Schröder B., Skidmore A.K., Zurell D., Lautenbach S., 2013. Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. Ecography 36: 27-46.
- Elith, J. and J.R. Leathwick, 2009. Species distribution models: Ecological explanation and prediction across space and time. Annual Review of Ecology, Evolution, and Systematics 40: 677-697.
- Elith, J., C.H. Graham, R.P. Anderson, M. Dudik, S. Ferrier, A. Guisan, R.J. Hijmans, F. Huettmann, J. Leathwick, A. Lehmann, J. Li, L.G. Lohmann, B. Loiselle, G. Manion, C. Moritz, M. Nakamura, Y. Nakazawa, J. McC. Overton, A.T. Peterson, S. Phillips, K. Richardson, R. Scachetti-Pereira, R. Schapire, J. Soberon, S. Williams, M. Wisz and N. Zimmerman, 2006. Novel methods improve prediction of species' distributions from occurrence data. Ecography 29: 129-151.
- Elith, J., S.J. Phillips, T. Hastie, M. Dudik, Y.E. Chee, C.J. Yates, 2011. A statistical explanation of MaxEnt for ecologists. Diversity and Distributions 17:43-57.
- Elith, J., J.R. Leathwick and T. Hastie, 2009. A working guide to boosted regression trees. Journal of Animal Ecology 77: 802-81
- Ferrier, S. and A. Guisan, 2006. Spatial modelling of biodiversity at the community level. Journal of Applied Ecology 43: 393-40

- Fielding, A.H. and J.F. Bell, 1997. A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation 24: 38-49
- Franklin, J. 2009. Mapping Species Distributions: Spatial Inference and Prediction. Cambridge University Press, Cambridge, UK.
- Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. The Annals of Statistics 29: 1189-1232.
- Graham, C.H., S. Ferrier, F. Huettman, C. Moritz and A. T Peterson, 2004. New developments in museum-based informatics and applications in biodiversity analysis. Trends in Ecology and Evolution 19: 497-503.
- Graham, C.H., J. Elith, R.J. Hijmans, A. Guisan, A.T. Peterson, B.A. Loiselle and the NCEAS Predicting Species Distributions Working Group, 2007. The influence of spatial errors in species occurrence data used in distribution models. Journal of Applied Ecology 45: 239-247
- Guisan, A., T.C. Edwards Jr, and T. Hastie, 2002. Generalized linear and generalized additive models in studies of species distributions: setting the scene. Ecological Modelling 157: 89-100.
- Guo, Q., M. Kelly, and C. Graham, 2005. Support vector machines for predicting distribution of Sudden Oak Death in California. Ecological Modeling 182: 75-90
- Guralnick, R.P., J. Wieczorek, R. Beaman, R.J. Hijmans and the BioGeomancer Working Group, 2006. BioGeomancer: Automated georeferencing to map the world's biodiversity data. PLoS Biology 4: 1908-1909.
- Hastie, T.J. and R.J. Tibshirani, 1990. Generalized Additive Models. Chapman & Hall/CRC.
- Hastie, T., R. Tibshirani and J. Friedman, 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Second Edition)
- Hijmans, R.J., 2012. Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null-model. Ecology 93: 679-688.
- Hijmans R.J., and C.H. Graham, 2006. Testing the ability of climate envelope models to predict the effect of climate change on species distributions. Global change biology 12: 2272-2281.
- Hijmans, R.J., M. Schreuder, J. de la Cruz and L. Guarino, 1999. Using GIS to check coordinates of germplasm accessions. Genetic Resources and Crop Evolution 46: 291-296.
- Hijmans, R.J., S.E. Cameron, J.L. Parra, P.G. Jones and A. Jarvis, 2005. Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology 25: 1965-1978.
- Jiménez-Valverde, A. 2011. Insights into the area under the receiver operating characteristic curve (AUC) as a discrimination measure in species distribution modelling. Global Ecology and Biogeography (on-line early): DOI: 10.1111/j.1466-8238.2011.00683.
- Karatzoglou, A., D. Meyer and K. Hornik, 2006. Support Vector Machines in R. Journal of statistical software 15(9).
- Kéry M., B. Gardner, and C. Monnerat, 2010. Predicting species distributions from checklist data using site-occupancy models. J. Biogeogr. 37: 1851–1862
- Lehmann, A., J. McC. Overton and J.R. Leathwick, 2002. GRASP: Generalized Regression Analysis and Spatial Predictions. Ecological Modelling 157: 189-207.
- Leathwick J., and D. Whitehead, 2001. Soil and atmospheric water deficits and the distribution of New Zealand's indigenous tree species. Functional Ecology 15: 233–242.
- Liu C., P.M. Berry, T.P. Dawson, and R.G. Pearson, 2005. Selecting thresholds of occurrence in the prediction of species distributions. Ecography 28: 385-393.
- Liu C., White M., Newell G., 2011. Measuring and comparing the accuracy of species distribution models with presence–absence data. Ecography 34: 232-243.

- Lobo, J.M. 2008. More complex distribution models or more representative data? Biodiversity Informatics 5: 14-19.
- Lobo, J.M., A. Jiménez-Valverde and R. Real, 2007. AUC: a misleading measure of the performance of predictive distribution models. Global Ecology and Biogeography 17: 145-151.
- Lozier, J.D., P. Aniello, and M.J. Hickerson, 2009. Predicting the distribution of Sasquatch in western North America: anything goes with ecological niche modelling. Journal of Biogeography 36: 1623–1627
- Mahalanobis, P.C., 1936. On the generalised distance in statistics. Proceedings of the National Institute of Sciences of India 2: 49-55.
- Mellert K.H., V. Fensterer, H. Küchenhoff, B. Reger, C. Kölling, H.J. Klemmt, and J. Ewald, 2011. Hypothesis-driven species distribution models for tree species in the Bavarian Alps. Journal of Vegetation Science 22: 635-646.
- Nix, H.A., 1986. A biogeographic analysis of Australian elapid snakes. In: Atlas of Elapid Snakes of Australia. (Ed.) R. Longmore, pp. 4-15. Australian Flora and Fauna Series Number 7. Australian Government Publishing Service: Canberra.
- Olson, D.M, E. Dinerstein, E.D. Wikramanayake, N.D. Burgess, G.V.N. Powell, E.C. Underwood, J.A. D'amico, I. Itoua, H.E. Strand, J.C. Morrison, C.J. Loucks, T.F. Allnutt, T.H. Ricketts, Y. Kura, J.F. Lamoreux, W.W. Wettengel, P. Hedao, and K.R. Kassem. 2001. Terrestrial Ecoregions of the World: A New Map of Life on Earth. BioScience 51: 933-938
- Peterson, A.T., J. Soberón, R.G. Pearson, R.P. Anderson, E. Martínez-Meyer, M. Nakamura and M.B. Araújo, 2011. Ecological Niches and Geographic Distributions. Monographs in Population Biology 49. Princeton University Press, 328p.
- Phillips S.J. and J. Elith, 2011. Logistic methods for resource selection functions and presence-only species distribution models, AAAI (Association for the Advancement of Artificial Intelligence), San Francisco, USA.
- Phillips, S.J., R.P. Anderson, R.E. Schapire, 2006. Maximum entropy modeling of species geographic distributions. Ecological Modelling 190: 231-259.
- Phillips, S.J., M. Dudik, J. Elith, C.H. Graham, A. Lehmann, J. Leathwick, and S. Ferrier. 2009. Sample selection bias and presence-only distribution models: implications for background and pseudo-absence data. Ecological Applications 19: 181-197.
- Potts J. and J. Elith, 2006. Comparing species abundance models. Ecological Modelling 199: 153-163.
- Thuiller, W. 2003. BIOMOD - optimizing predictions of species distributions and projecting potential future shifts under global change. Global Change Biology 9: 1353-1362.
- Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.
- VanDerWal J., L.P. Shoo, C. Graham and S.E. Williams, 2009. Selecting pseudo-absence data for presence-only distribution modeling: how far should you stray from what you know? Ecological Modelling 220: 589-594.
- Ward G., T. Hastie, S.C. Barry, J. Elith and J.R. Leathwick, 2009. Presence-only data and the EM algorithm. Biometrics 65: 554-563.
- Wieczorek, J., Q. Guo and R.J. Hijmans, 2004. The point-radius method for georeferencing point localities and calculating associated uncertainty. International Journal of Geographic Information Science 18: 745-767.
- Wisz, M.S., R.J. Hijmans, J. Li, A.T. Peterson, C.H. Graham, A. Guisan, and the NCEAS Predicting Species Distributions Working Group, 2008. Effects of sample size on the performance of species distribution models. Diversity and Distributions 14: 763-773.
- Wood, S., 2006. Generalized Additive Models: An Introduction with *R*. Chapman & Hall/CRC.

CHAPTER

NINE

APPENDIX: BOOSTED REGRESSION TREES FOR ECOLOGICAL MODELING

Jane Elith and John Leathwick

9.1 Introduction

This is a brief tutorial to accompany a set of functions that we have written to facilitate fitting BRT (boosted regression tree) models in *R*. This tutorial is a modified version of the tutorial accompanying Elith, Leathwick and Hastie's article in Journal of Animal Ecology. It has been adjusted to match the implementation of these functions in the 'dismo' package. The gbm* functions in the dismo package extend functions in the 'gbm' package by Greg Ridgeway. The goal of our functions is to make the functions in the 'gbm' package easier to apply to ecological data, and to enhance interpretation.

The tutorial is aimed at helping you to learn the mechanics of how to use the functions and to develop a BRT model in *R*. It does not explain what a BRT model is - for that, see the references at the end of the tutorial, and the documentation of the gbm package. For an example application with similar data as in this tutorial, see Elith et al., 2008.

The gbm functions in 'dismo' are as follows:

1. `gbm.step` - Fits a gbm model to one or more response variables, using cross-validation to estimate the optimal number of trees. This requires use of the utility functions roc, calibration and calc.deviance.
2. `gbm.fixed`, `gbm.holdout` - Alternative functions for fitting gbm models, implementing options provided in the gbm package.
3. `gbm.simplify` - Code to perform backwards elimination of variables, to drop those that give no evidence of improving predictive performance.
4. `gbm.plot` - Plots the partial dependence of the response on one or more predictors.
5. `gbm.plot.fits` - Plots the fitted values from a gbm object returned by any of the model fitting options. This can give a more reliable guide to the shape of the fitted surface than can be obtained from the individual functions, particularly when predictor variables are correlated and/or samples are unevenly distributed in environmental space.
6. `gbm.interactions` - Tests whether interactions have been detected and modelled, and reports the relative strength of these. Results can be visualised with gbm.perspec

9.2 Example data

Two sets of presence/absence data for *Anguilla australis* (Angaus) are available. One for model training (building) and one for model testing (evaluation). In the example below we load the training data. Presence (1) and absence (0) is recorded in column 2. The environmental variables are in columns 3 to 14. This is the same data as used in Elith, Leathwick and Hastie (2008).

```
library(dismo)
data(Anguilla_train)
head(Anguilla_train)
##   Site Angaus SegSumT SegTSeas SegLowFlow DSDist DSMaxSlope USAvgT USRainDays
## 1    1      0    16.0    -0.10      1.036  50.20       0.57   0.09      2.470
## 2    2      1    18.7     1.51      1.003 132.53       1.15   0.20      1.153
## 3    3      0    18.3     0.37      1.001 107.44       0.57   0.49      0.847
## 4    4      0    16.7    -3.80      1.000 166.82       1.72   0.90      0.210
## 5    5      1    17.2     0.33      1.005   3.95       1.15  -1.20      1.980
## 6    6      0    15.1     1.83      1.015  11.17       1.72  -0.20      3.300
##   USSlope USNative DSDam   Method LocSed
## 1     9.8     0.81     0 electric    4.8
## 2     8.3     0.34     0 electric    2.0
## 3     0.4     0.00     0      spo    1.0
## 4     0.4     0.22     1 electric    4.0
## 5    21.9     0.96     0 electric    4.7
## 6    25.7     1.00     0 electric    4.5
```

9.3 Fitting a model

To fit a gbm model, you need to decide what settings to use the article associated with this tutorial gives you information on what to use as rules of thumb. These data have 1000 sites, comprising 202 presence records for the short-finned eel (the command `sum(model.data$Angaus)` will give you the total number of presences). As a first guess you could decide:

1. There are enough data to model interactions of reasonable complexity
2. A lr of about 0.01 could be a reasonable starting point.

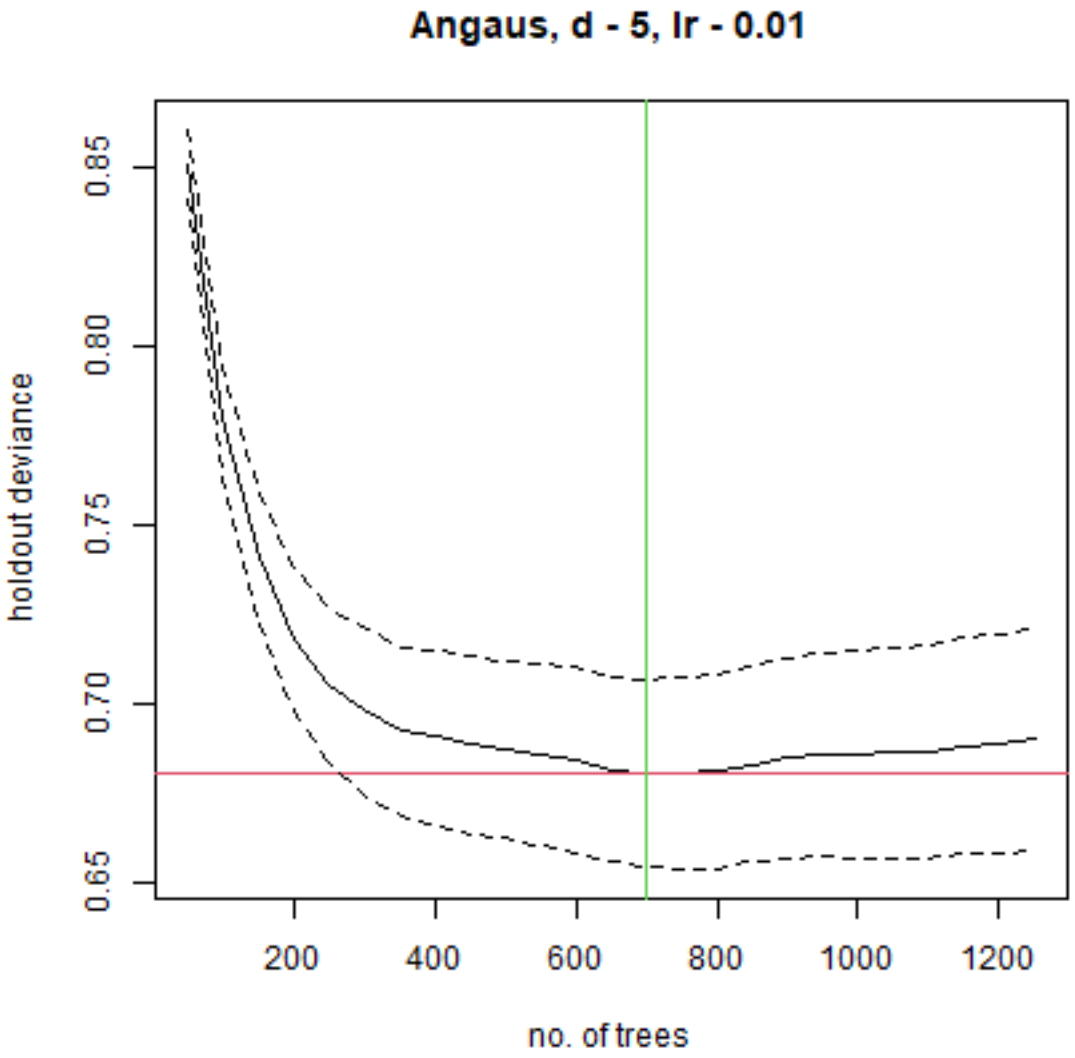
To below example shows how to use our function that steps forward and identifies the optimal number of trees (nt).

```
angaus.tc5.lr01 <- gbm.step(data=Anguilla_train, gbm.x = 3:13, gbm.y = 2,
                        family = "bernoulli", tree.complexity = 5,
                        learning.rate = 0.01, bag.fraction = 0.5)
##
##
##  GBM STEP - version 2.9
##
## Performing cross-validation optimisation of a boosted regression tree model
## for Angaus and using a family of bernoulli
## Using 1000 observations and 11 predictors
## creating 10 initial models of 50 trees
##
##  folds are stratified by prevalence
## total mean deviance =  1.0063
```

(continues on next page)

(continued from previous page)

```
## tolerance is fixed at  0.001
## ntrees resid. dev.
## 50    0.8506
## now adding trees...
## 100   0.7798
## 150   0.7409
## 200   0.7182
## 250   0.705
## 300   0.6983
## 350   0.6925
## 400   0.6908
## 450   0.6888
## 500   0.6875
## 550   0.686
## 600   0.6844
## 650   0.6819
## 700   0.6806
## 750   0.6809
## 800   0.6813
## 850   0.6833
## 900   0.685
## 950   0.6859
## 1000   0.6859
## 1050   0.6864
## 1100   0.687
## 1150   0.6885
## 1200   0.6891
## 1250   0.6904
## fitting final gbm model with a fixed number of 700 trees for Angaus
```




```
##
## mean total deviance = 1.006
## mean residual deviance = 0.439
##
## estimated cv deviance = 0.681 ; se = 0.026
##
## training data correlation = 0.796
## cv correlation =  0.579 ; se = 0.026
##
## training data AUC score = 0.965
## cv AUC score = 0.871 ; se = 0.01
##
## elapsed time -  0.2 minutes
```

Above we used the function gbm.step, this function is an alternative to the cross-validation provided in the gbm package.

We have passed information to the function about data and settings. We have defined:

the data.frame containing the data is `Anguilla_train`;

the predictor variables - `gbm.x = c(3:13)` - which we do using a vector consisting of the indices for the data columns containing the predictors (i.e., here the predictors are columns 3 to 13 in Anguilla_train);

the response variable - `gbm.y = 2` - indicating the column number for the species (response) data;

the nature of the error structure - For example, `family = 'bernoulli'` (note the quotes);

the tree complexity - we are trying a tree complexity of 5 for a start;

the learning rate - we are trying with 0.01;

the bag fraction - our default is 0.75; here we are using 0.5;

Everything else - that is, all the other things that we could change if we wanted to (see the help file, and the documentation of the gbm package) - are set at their defaults if they are not named in the call. If you want to see what else you could change, you can type gbm.step and all the code will write itself to screen, or type args(gbm.step) and it will open in an editor window.

Running a model such as that described above writes progress reports to the screen, makes a graph, and returns an object containing a number of components. Firstly, the things you can see: The R console will show something like this (not identical, because remember that these models are stochastic and therefore slightly different each time you run them, unless you set the seed or make them deterministic by using a bag fraction of 1)

This reports a brief model summary. All these values are also retained in the model object, so they will be permanently kept (as long as you save the R workspace before quitting).

This model was built with the default 10-fold cross-validation. The solid black curve is the mean, and the dotted curves about 1 standard error, for the changes in predictive deviance (ie as measured on the excluded folds of the cross-validation). The red line shows the minimum of the mean, and the green line the number of trees at which that occurs. The final model that is returned in the model object is built on the full data set, using the number of trees identified as optimal.

The returned object is a list (see R documentation if you don't know what that is), and the names of the components can be seen by typing:

To pull out one component of the list, use a number (angaus.tc5.lr01[[29]]) or name (angaus.tc5.lr01$cv.statistics) - but be careful, some are as big as the dataset, e.g. there will be 1000 fitted values. Find this by typing

```
length(angaus.tc5.lr01$fitted)
```

The way we organise our functions is to return exactly what Ridgeway's function in the gbm package returned, plus extra things that are relevant to our code. You will see by looking at the final parts of the gbm.step code that we have added components 25 onwards, that is, from gbm.call on. See the gbm documentation for what his parts comprise. Ours are:

gbm.call - A list containing the details of the original call to gbm.step

fitted - The fitted values from the final tree, on the response scale

fitted.vars - The variance of the fitted values, on the response scale

residuals - The residuals for the fitted values, on the response scale

contributions - The relative importance of the variables, produced from the gbm summary function

self.statistics - The relevant set of evaluation statistics, calculated on the fitted values - i.e. this is only interesting in so far as it demonstrates "evaluation" (i.e. fit) on the training data. It should NOT be reported as the model predictive performance.

cv.statistics These are the most appropriate evaluation statistics. We calculate each statistic within each fold (at the identified optimal number of trees that is calculated on the mean change in predictive deviance over all folds), then present here the mean and standard error of those fold-based statistics.

weights - the weights used in fitting the model (by default, "1" for each observation - i.e. equal weights).

trees.fitted - A record of the number of trees fitted at each step in the stagewise fitting; only relevant for later calculations

training.loss.values - The stagewise changes in deviance on the training data

cv.values - the mean of the CV estimates of predictive deviance, calculated at each step in the stagewise process - this and the next are used in the plot shown above

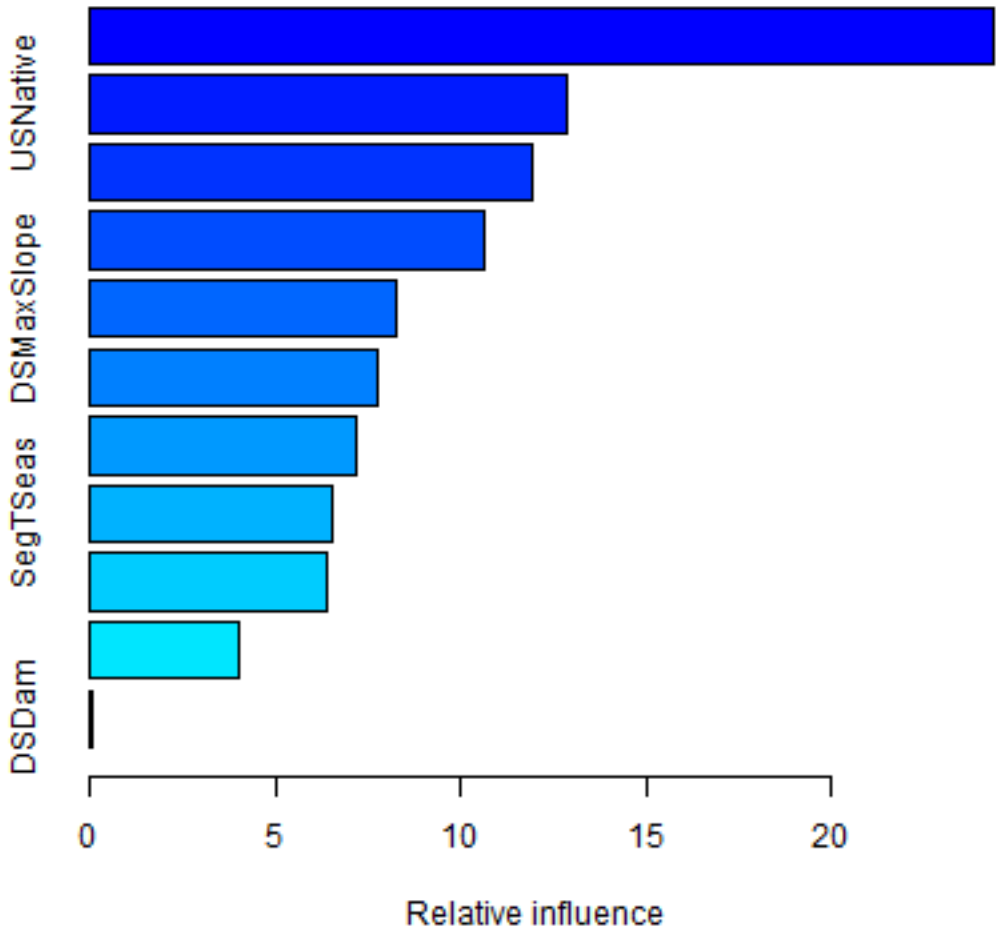
cv.loss.ses - standard errors in CV estimates of predictive deviance at each step in the stagewise process

cv.loss.matrix - the matrix of values from which cv.values were calculated - as many rows as folds in the CV

cv.roc.matrix - as above, but the values in it are area under the curve estimated on the excluded data, instead of deviance in the cv.loss.matrix.

You can look at variable importance with the summary function

```
names(angaus.tc5.lr01)
##  [1] "initF"                "fit"                  "train.error"
##  [4] "valid.error"          "oobag.improve"        "trees"
##  [7] "c.splits"             "bag.fraction"         "distribution"
## [10] "interaction.depth"    "n.minobsinnode"       "num.classes"
## [13] "n.trees"              "nTrain"               "train.fraction"
## [16] "response.name"        "shrinkage"            "var.levels"
## [19] "var.monotone"         "var.names"            "var.type"
## [22] "verbose"              "data"                 "Terms"
## [25] "cv.folds"             "call"                 "m"
## [28] "gbm.call"             "fitted"               "fitted.vars"
## [31] "residuals"            "contributions"        "self.statistics"
## [34] "cv.statistics"        "weights"              "trees.fitted"
## [37] "training.loss.values" "cv.values"            "cv.loss.ses"
## [40] "cv.loss.matrix"       "cv.roc.matrix"
summary(angaus.tc5.lr01)
```




```
##                   var     rel.inf
## SegSumT       SegSumT 24.38814078
## USNative     USNative 12.87889951
## DSDist         DSDist 11.90091735
## Method         Method 10.61370646
## DSMaxSlope DSMaxSlope  8.23624294
## USRainDays USRainDays  7.77965715
## USSlope       USSlope  7.17173978
## SegTSeas     SegTSeas  6.51684705
## USAvgT         USAvgT  6.36887470
## SegLowFlow SegLowFlow  4.05900034
## DSDam           DSDam  0.08597395
```

9.4 Choosing the settings

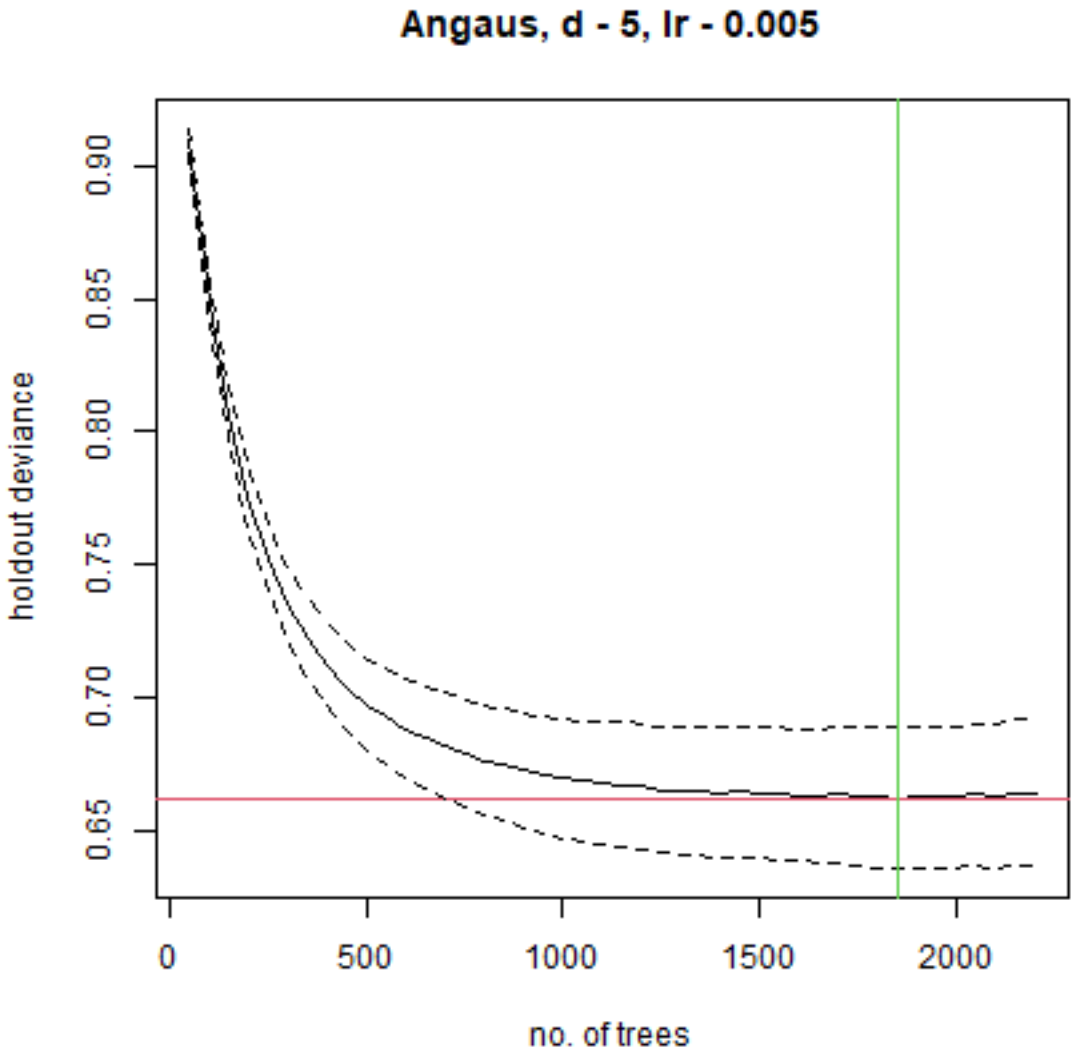
The above was a first guess at settings, using rules of thumb discussed in Elith et al. (2008). It made a model with only 650 trees, so our next step would be to reduce the lr. For example, try lr = 0.005, to aim for over 1000 trees:

```
angaus.tc5.lr005 <- gbm.step(data=Anguilla_train, gbm.x = 3:13, gbm.y = 2,
                      family = "bernoulli", tree.complexity = 5,
                      learning.rate = 0.005, bag.fraction = 0.5)
##
##
##  GBM STEP - version 2.9
##
## Performing cross-validation optimisation of a boosted regression tree model
## for Angaus and using a family of bernoulli
## Using 1000 observations and 11 predictors
## creating 10 initial models of 50 trees
##
##  folds are stratified by prevalence
## total mean deviance =  1.0063
## tolerance is fixed at  0.001
## ntrees resid. dev.
## 50    0.9092
## now adding trees...
## 100   0.8483
## 150   0.8069
## 200   0.7758
## 250   0.7527
## 300   0.7355
## 350   0.7224
## 400   0.7123
## 450   0.7043
## 500   0.6972
## 550   0.6929
## 600   0.6886
## 650   0.6853
## 700   0.6824
## 750   0.6796
## 800   0.6766
## 850   0.6748
## 900   0.6728
## 950   0.6713
## 1000   0.6697
## 1050   0.6688
## 1100   0.6679
## 1150   0.6675
## 1200   0.6669
## 1250   0.6654
## 1300   0.6655
## 1350   0.6648
## 1400   0.6645
## 1450   0.6648
## 1500   0.6643
## 1550   0.6642
```

(continues on next page)

(continued from previous page)

```
## 1600   0.6636
## 1650   0.6633
## 1700   0.6639
## 1750   0.6632
## 1800   0.6628
## 1850   0.6624
## 1900   0.663
## 1950   0.6627
## 2000   0.6629
## 2050   0.6637
## 2100   0.6635
## 2150   0.6645
## 2200   0.6643
## fitting final gbm model with a fixed number of 1850 trees for Angaus
```



```
##
## mean total deviance = 1.006
## mean residual deviance = 0.388
##
## estimated cv deviance = 0.662 ; se = 0.026
##
## training data correlation = 0.832
## cv correlation =  0.597 ; se = 0.019
##
## training data AUC score = 0.976
## cv AUC score = 0.879 ; se = 0.014
##
## elapsed time -  0.38 minutes
```

To more broadly explore whether other settings perform better, and assuming that these are the only data available, you could either split the data into a training and testing set or use the cross-validation results. You could systematically alter tc, lr and the bag fraction and compare the results. See the later section on prediction to find out how to predict to independent data and calculate relevant statistics.

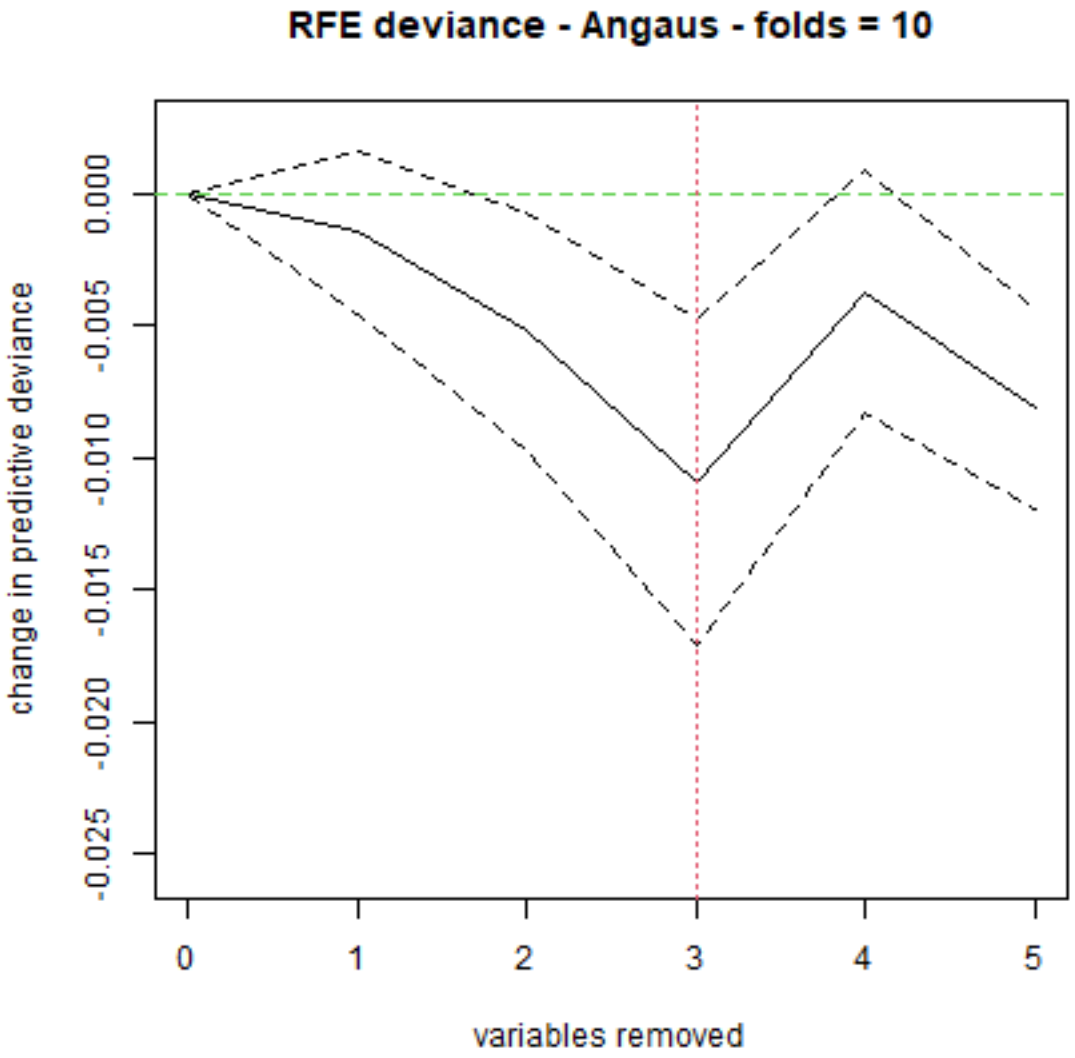
9.5 Alternative ways to fit models

The step function above is slower than just fitting one model and finding a minimum. If this is a problem, you could use our gbm.holdout code - this combines from the gbm package in ways we find useful. We tend to prefer gbm.step, especially when modelling many species, because it automatically finds the optimal number of trees. Alternatively, the gbm.fixed code allows you to fit a model of a set number of trees; this can be used, as in Elith et al. (2008), to predict to new data (see later section).

9.6 section{Simplifying the model

For a discussion of simplification see Appendix 2 of the online supplement to Elith et al (2008). Simplification builds many models, so it can be slow. For example, the code below took a few minutes to run on a modern laptop. In it we assess the value in simplifying the model built with a lr of 0.005, but only test dropping up to 5 variables (the "n.drop" argument; the default is an automatic rule so it continues until the average change in predictive deviance exceeds its original standard error as calculated in gbm.step).

```
angaus.simp <- gbm.simplify(angaus.tc5.lr005, n.drops = 5)
## gbm.simplify - version 2.9
## simplifying gbm.step model for Angaus with 11 predictors and 1000 observations
## original deviance = 0.6624(0.0263)
## a fixed number of 5 drops will be tested
## creating initial models...
## dropping predictor: 1 2 3 4 5
## processing final dropping of variables with full data
## 1-DSDam
## 2-SegLowFlow
## 3-USAvgT
## 4-SegTSeas
## 5-DSMaxSlope
```



For our run, this estimated that the optimal number of variables to drop was 1; yours could be slightly different:

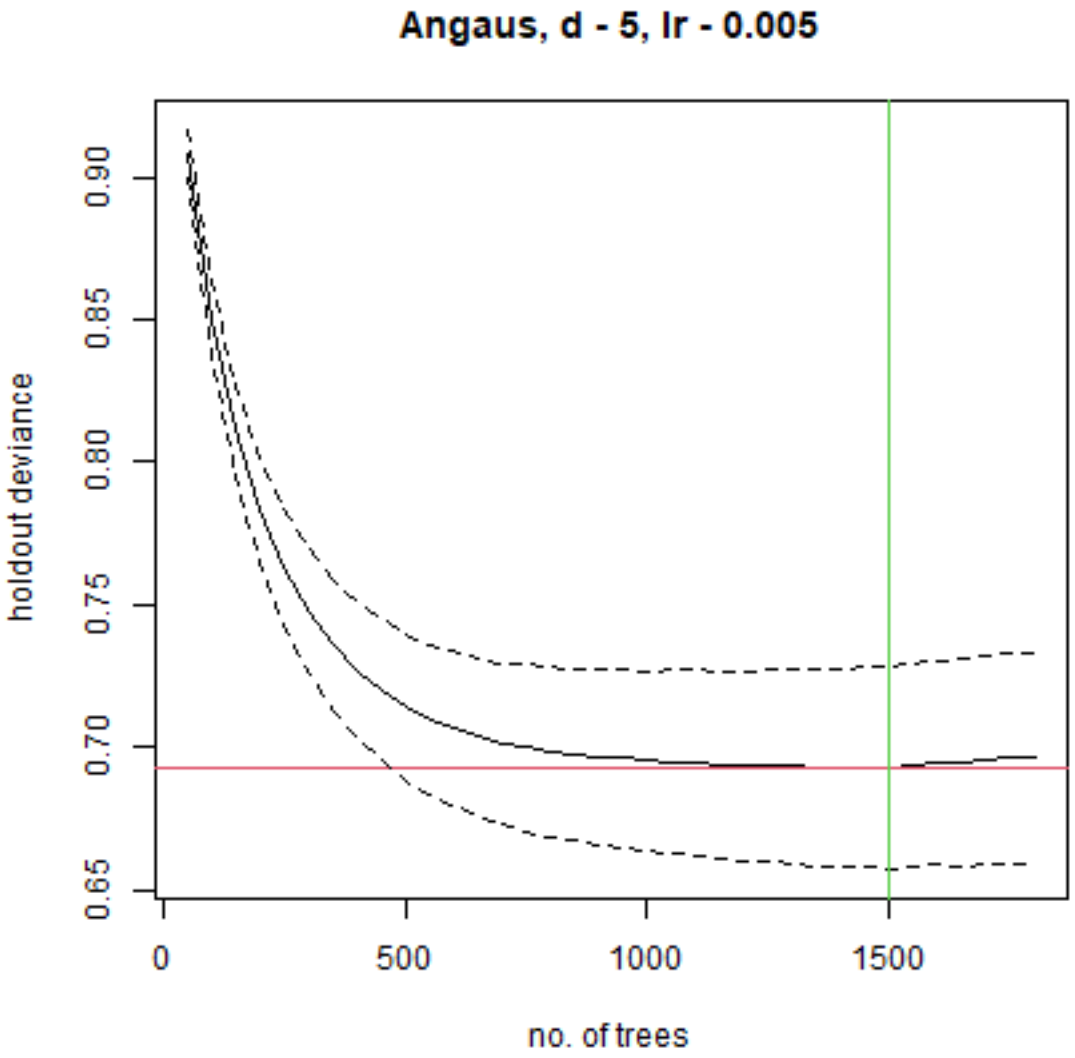
You can use the number indicated by the red vertical line, or look at the results in the angaus.simp object Now make a model with 1 predictor dropped, by indicating to the gbm.step call the relevant number of predictor(s) from the predictor list in the angaus.simp object - see highlights, below, in which we indicate we want to drop 1 variable by calling the second vector of predictor columns in the pred list, using `[[1]]`:

```
angaus.tc5.lr005.simp <- gbm.step(Anguilla_train,
                    gbm.x=angaus.simp$pred.list[[1]], gbm.y=2,
                    tree.complexity=5, learning.rate=0.005)
##
##
##  GBM STEP - version 2.9
##
## Performing cross-validation optimisation of a boosted regression tree model
## for Angaus and using a family of bernoulli
## Using 1000 observations and 10 predictors
```

(continues on next page)

(continued from previous page)

```
## creating 10 initial models of 50 trees
##
##  folds are stratified by prevalence
## total mean deviance =  1.0063
## tolerance is fixed at  0.001
## ntrees resid. dev.
## 50    0.9082
## now adding trees...
## 100   0.8502
## 150   0.8115
## 200   0.7834
## 250   0.7627
## 300   0.7472
## 350   0.7355
## 400   0.7268
## 450   0.72
## 500   0.7139
## 550   0.7099
## 600   0.7065
## 650   0.7037
## 700   0.7011
## 750   0.6998
## 800   0.6986
## 850   0.6974
## 900   0.6969
## 950   0.6961
## 1000   0.695
## 1050   0.6948
## 1100   0.6947
## 1150   0.6936
## 1200   0.6934
## 1250   0.6937
## 1300   0.6936
## 1350   0.693
## 1400   0.6927
## 1450   0.693
## 1500   0.6925
## 1550   0.6935
## 1600   0.6947
## 1650   0.6947
## 1700   0.6953
## 1750   0.696
## 1800   0.6962
## fitting final gbm model with a fixed number of 1500 trees for Angaus
```



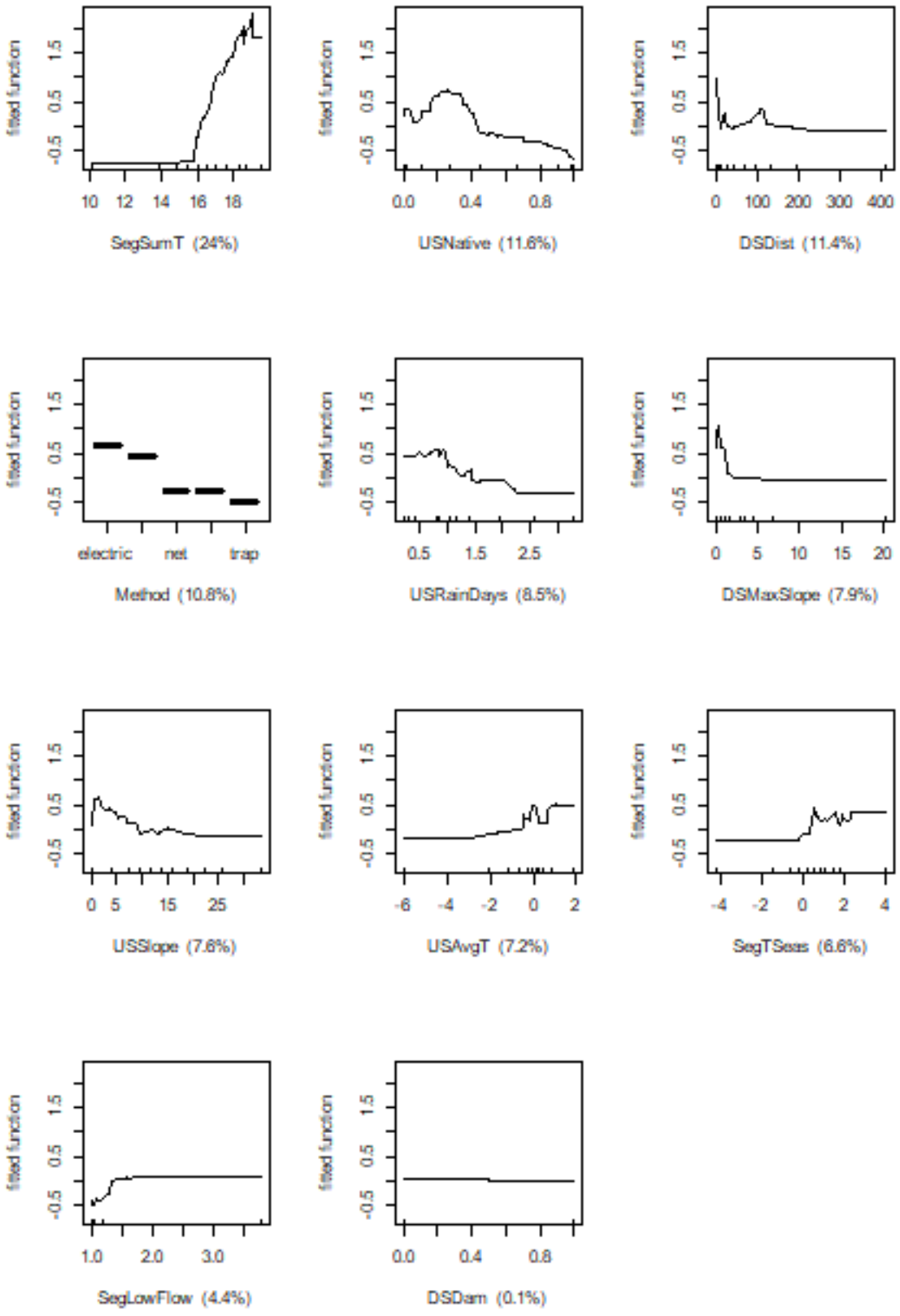

```
##
## mean total deviance = 1.006
## mean residual deviance = 0.419
##
## estimated cv deviance = 0.692 ; se = 0.035
##
## training data correlation = 0.813
## cv correlation =  0.574 ; se = 0.035
##
## training data AUC score = 0.971
## cv AUC score = 0.871 ; se = 0.012
##
## elapsed time -  0.28 minutes
```

This has now made a new model (angaus.tc5.lr005.simp) with the same list components as described earlier. We could continue to use it, but given that we don't particularly want a more simple model (our view is that, in a dataset of this size, included variables that contribute little are acceptable), we won't use it further.

9.7 Plotting the functions and fitted values from the model

The fitted functions from a BRT model created from any of our functions can be plotted using gbm.plot. If you want to plot all variables on one sheet first set up a graphics device with the right set-up - here we will make one with 3 rows and 4 columns:

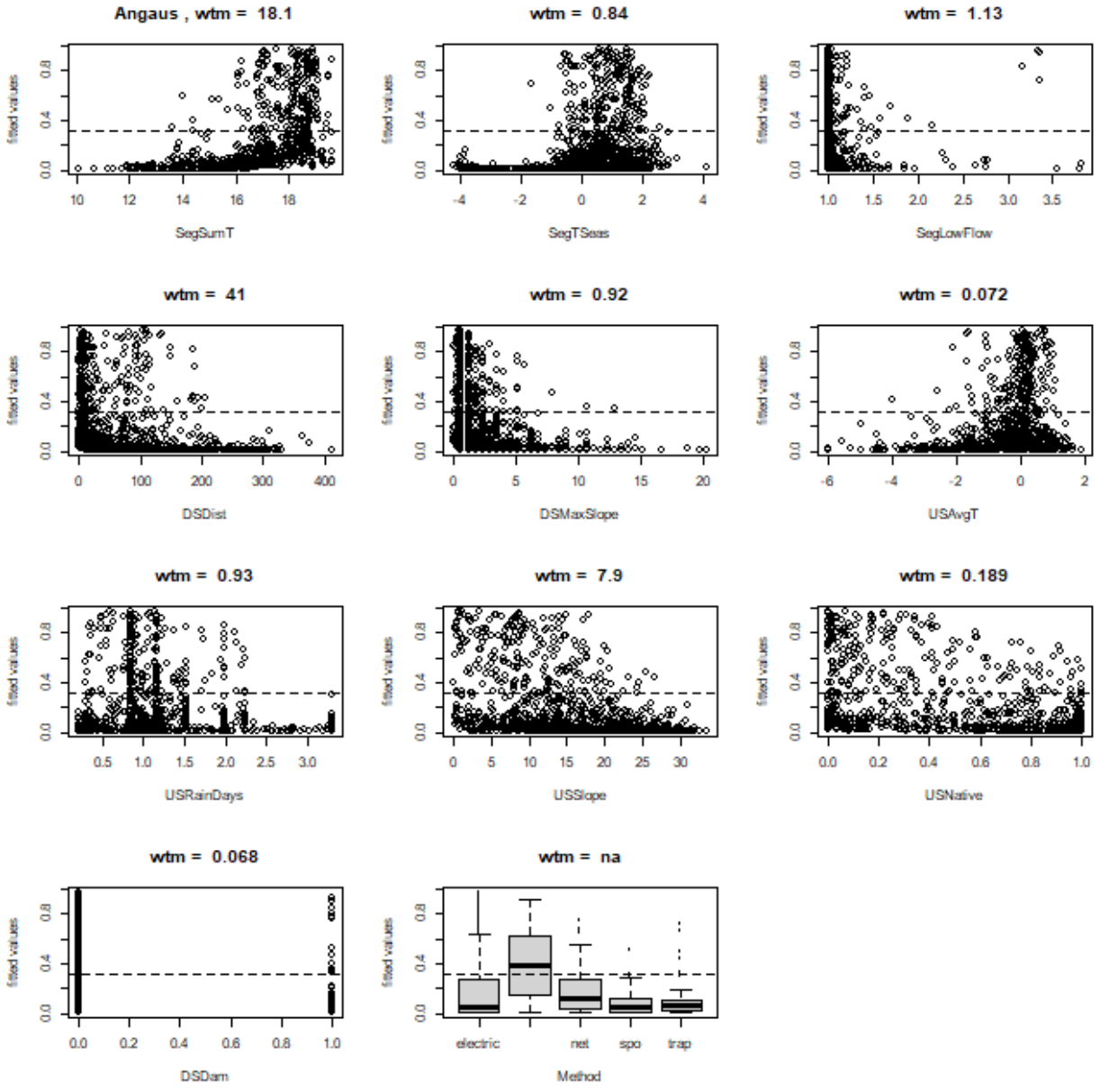
```
gbm.plot(angaus.tc5.lr005, n.plots=11, plot.layout=c(4, 3), write.title = FALSE)
```


fitted function
-0.5
0.5
1.5
10 12 14 16 18
SegSumT (24%)
0.0 0.4 0.8
USNative (11.6%)
0 100 200 300 400
DSDist (11.4%)
electric net trap
Method (10.8%)
0.5 1.5 2.5
USRainDays (8.5%)
0 5 10 15 20
DSMaxSlope (7.9%)
0 5 15 25
USSlope (7.6%)
-6 -4 -2 0 2
USAvgT (7.2%)
-4 -2 0 2 4
SegTSeas (6.6%)
1.0 2.0 3.0
SegLowFlow (4.4%)
0.0 0.4 0.8
DSDam (0.1%)

Additional arguments to this function allow for making a smoothed representation of the plot, allowing different vertical scales for each variable, omitting (and formatting) the rugs, and plotting a single variable.

Depending on the distribution of observations within the environmental space, fitted functions can give a misleading indication about the distribution of the fitted values in relation to each predictor. The function gbm.plot.fits has been provided to plot the fitted values in relation to each of the predictors used in the model.

```
gbm.plot.fits(angaus.tc5.lr005)
```



This has options that allow for the plotting of all fitted values or of fitted values only for positive observations, or the plotting of fitted values in factor type graphs that are much quicker to print. Values above each graph indicate the weighted mean of fitted values in relation to each non-factor predictor.

9.8 Interrogate and plot the interactions

This code assesses the extent to which pairwise interactions exist in the data.

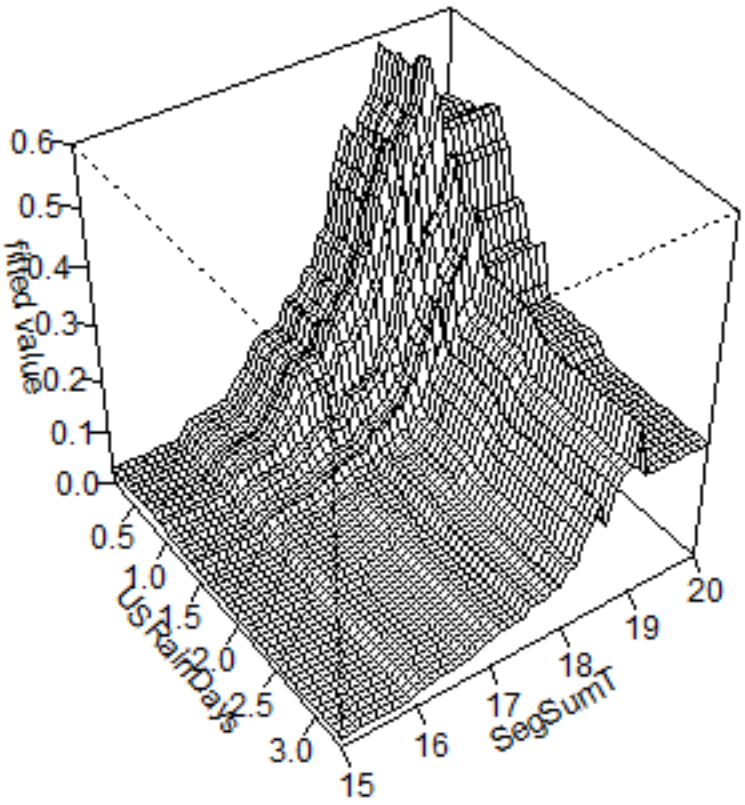
`find.int <- gbm.interactions(angaus.tc5.lr005)`.

The returned object, here named test.int, is a list. The first 2 components summarise the results, first as a ranked list of the 5 most important pairwise interactions, and the second tabulating all pairwise interactions. The variable index numbers in `rank.list can be used for plotting.

```
find.int <- gbm.interactions(angaus.tc5.lr005)
## gbm.interactions - version 2.9
## Cross tabulating interactions for gbm model with 11 predictors
## 1 2 3 4 5 6 7 8 9 10
find.int$interactions
##            SegSumT SegTSeas SegLowFlow DSDist DSMaxSlope USAvgT USRainDays
## SegSumT          0     4.21       0.20  21.00       1.49  12.90       25.25
## SegTSeas         0     0.00       1.88   9.15       1.29   0.52        1.35
## SegLowFlow       0     0.00       0.00   0.51       0.75   1.65        0.11
## DSDist           0     0.00       0.00   0.00       0.06   0.11        4.88
## DSMaxSlope       0     0.00       0.00   0.00       0.00   1.33        0.38
## USAvgT           0     0.00       0.00   0.00       0.00   0.00        0.47
## USRainDays       0     0.00       0.00   0.00       0.00   0.00        0.00
## USSlope          0     0.00       0.00   0.00       0.00   0.00        0.00
## USNative         0     0.00       0.00   0.00       0.00   0.00        0.00
## DSDam            0     0.00       0.00   0.00       0.00   0.00        0.00
## Method           0     0.00       0.00   0.00       0.00   0.00        0.00
##            USSlope USNative DSDam Method
## SegSumT       4.03     7.39  0.04  13.87
## SegTSeas      0.24     0.67  0.00   1.46
## SegLowFlow    1.35     0.36  0.00   0.82
## DSDist        1.07     1.47  0.00   2.33
## DSMaxSlope    0.86     2.88  0.00   0.39
## USAvgT        0.19     2.31  0.00   0.82
## USRainDays    0.46     3.67  0.02   2.78
## USSlope       0.00     2.00  0.00   3.82
## USNative      0.00     0.00  0.00   2.88
## DSDam         0.00     0.00  0.00   0.00
## Method        0.00     0.00  0.00   0.00
find.int$rank.list
##   var1.index var1.names var2.index var2.names int.size
## 1          7 USRainDays          1    SegSumT    25.25
## 2          4     DSDist          1    SegSumT    21.00
## 3         11     Method          1    SegSumT    13.87
## 4          6     USAvgT          1    SegSumT    12.90
## 5          4     DSDist          2   SegTSeas     9.15
## 6          9   USNative          1    SegSumT     7.39
```

You can plot pairwise interactions like this:

```
gbm.perspec(angaus.tc5.lr005, 7, 1, y.range=c(15,20), z.range=c(0,0.6))
## maximum value = 0.62
## Warning in persp.default(x = x.var, y = y.var, z = pred.matrix, zlim =
## z.range, : surface extends beyond the box
```



Additional options allow specifications of label names, rotations of the 3D graph and so on.

9.9 Predicting to new data

If you want to predict to a set of sites (rather than to a whole map), the general procedure is to set up a data.frame with rows for sites and columns for the variables that are in your model. R is case sensitive; the names need to exactly match those in the model. Other columns such as site IDs etc can also exist in the data.frame (and are ignored).

Our dataset for predicting to sites is in a file called Anguilla_test. The "Method" column needs to be converted to a factor, with levels matching those in the modelling data. To make predictions to sites from the BRT model use predict (or predict.gbm) from the gbm package

The predictions are in a vector called preds. These are evaluation sites, and have observations in column 1 (named Angaus_obs).

They are independent of the model building set and were used for an evaluation with independent data. Note that the calc.deviance function has different formulae for different distributions of data; the default is binomial, so we didn't

specify it in the call

```
data(Anguilla_test)
library(gbm)
preds <- predict.gbm(angaus.tc5.lr005, Anguilla_test,
         n.trees=angaus.tc5.lr005$gbm.call$best.trees, type="response")

calc.deviance(obs=Anguilla_test$Angaus_obs, pred=preds, calc.mean=TRUE)
## [1] 0.7420087
d <- cbind(Anguilla_test$Angaus_obs, preds)
pres <- d[d[,1]==1, 2]
abs <- d[d[,1]==0, 2]
e <- evaluate(p=pres, a=abs)
e
## class          : ModelEvaluation
## n presences    : 107
## n absences     : 393
## AUC            : 0.8610972
## cor            : 0.5263345
## max TPR+TNR at : 0.1021038
```

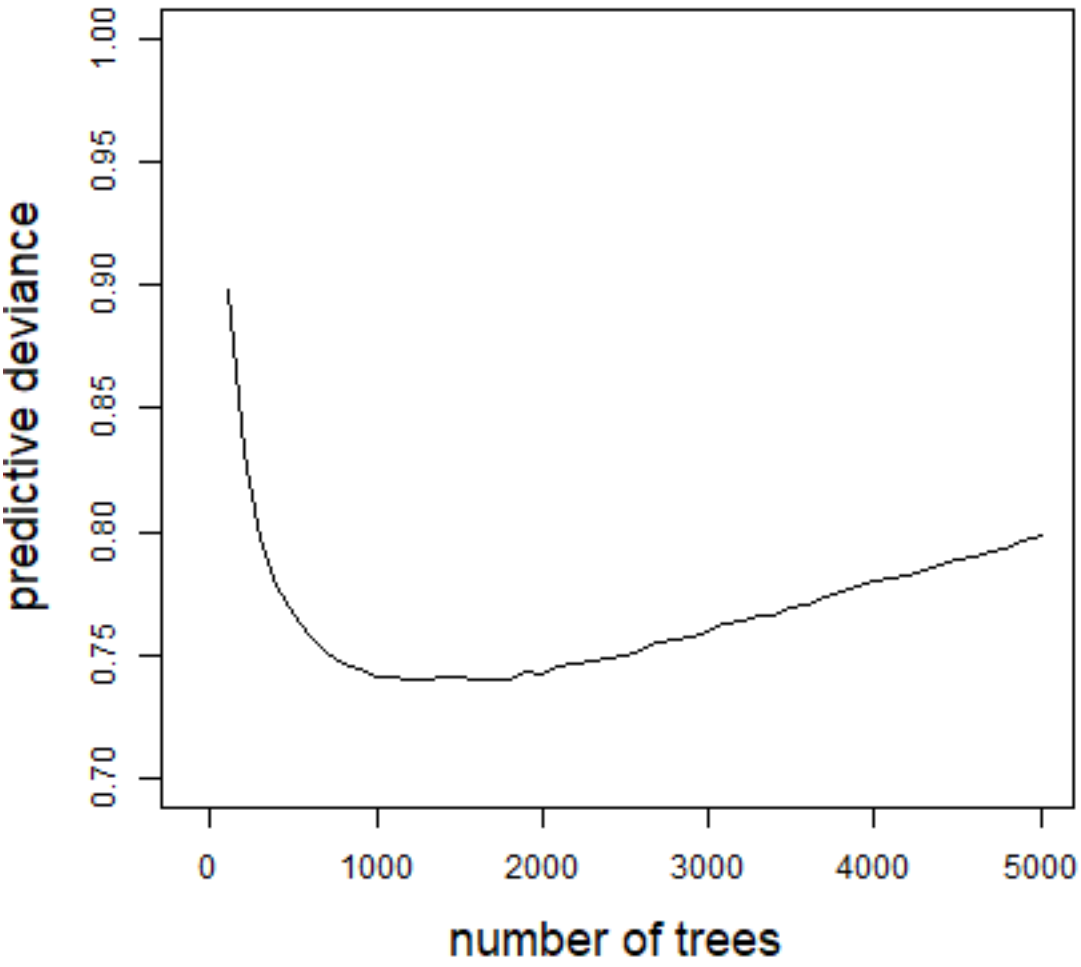
One useful feature of prediction in gbm is you can predict to a varying number of trees. See the highlighted code below to how to predict to a vector of trees. The full set of code here shows how to make one of the graphed lines from Fig. 2 in our paper, using a model of 5000 trees developed with gbm.fixed

```
angaus.5000 <- gbm.fixed(data=Anguilla_train, gbm.x=3:13, gbm.y=2,
               learning.rate=0.005, tree.complexity=5, n.trees=5000)
## [1] fitting gbm model with a fixed number of 5000 trees for Angaus
## [1] total deviance = 1006.33
## [1] residual deviance = 203.21
tree.list <- seq(100, 5000, by=100)
pred <- predict.gbm(angaus.5000, Anguilla_test, n.trees=tree.list, "response")
```

Note that the code above makes a matrix, with each column being the predictions from the model angaus.5000 to the number of trees specified by that element of tree.list - for example, the predictions in column 5 are for tree.list[5] = 500 trees. Now to calculate the deviance of all these results, and plot them:

```
angaus.pred.deviance <- rep(0,50)
for (i in 1:50) {
   angaus.pred.deviance[i] <- calc.deviance(Anguilla_test$Angaus_obs,
                              pred[,i], calc.mean=TRUE)
}
```

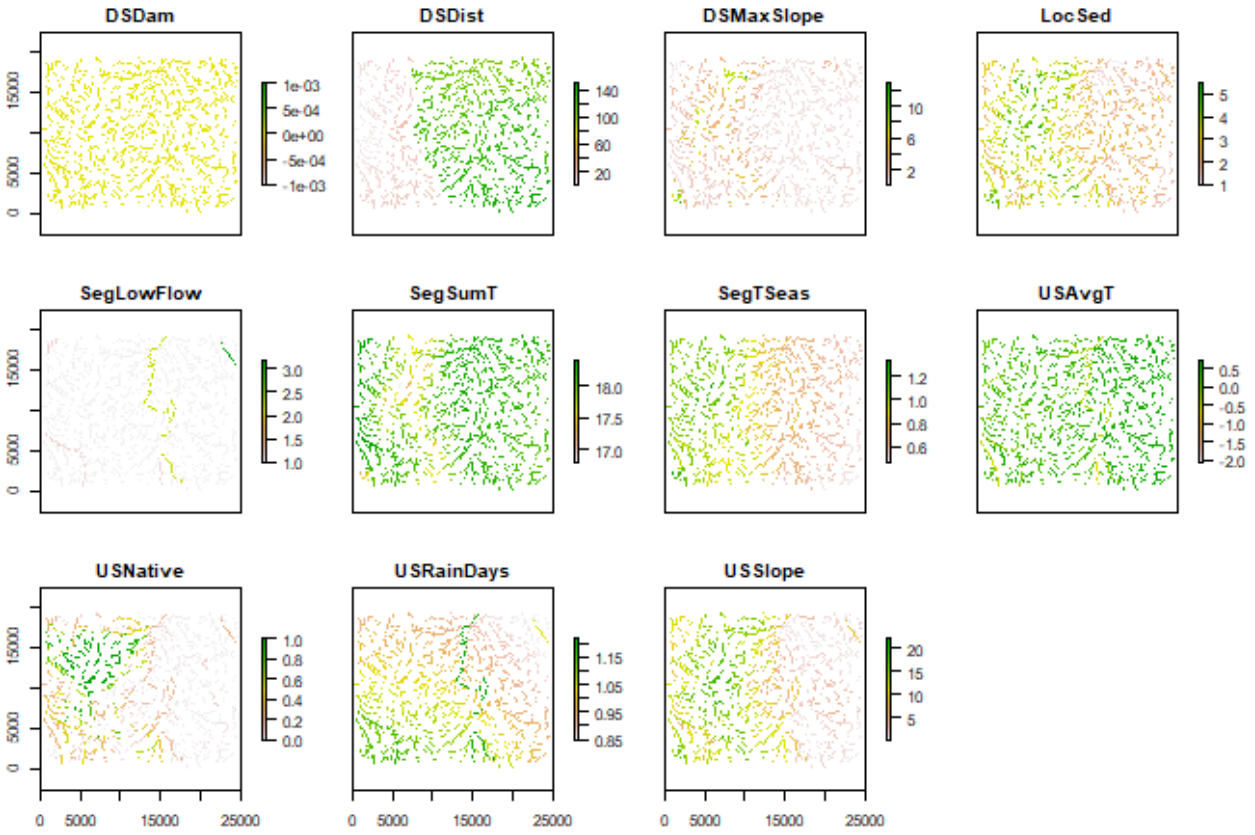
```
plot(tree.list, angaus.pred.deviance, ylim=c(0.7,1), xlim=c(-100,5000),
     type='l', xlab="number of trees", ylab="predictive deviance",
     cex.lab=1.5)
```



9.10 Spatial prediction

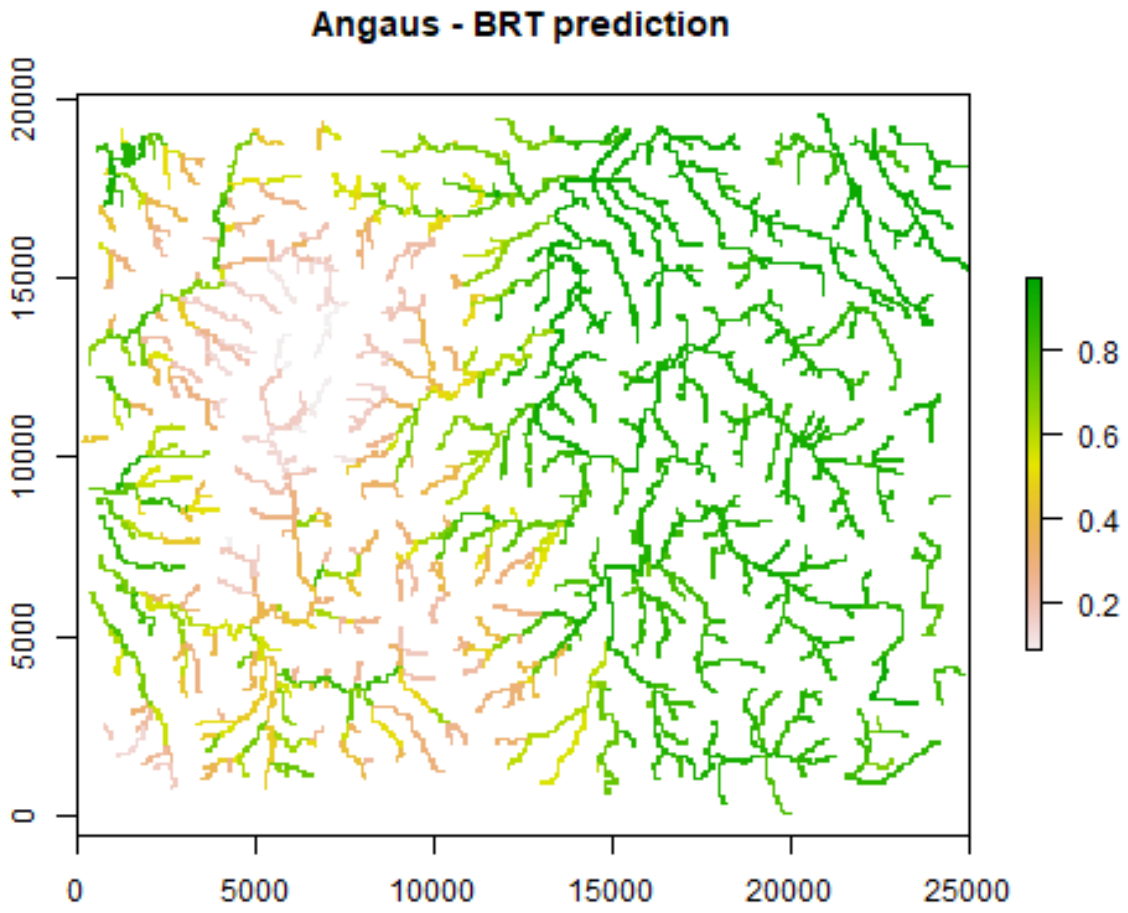
Here we show how to predict to a whole map (technically to a RasterLayer object) using the predict version in the `raster` package. The predictor variables are available as a RasterBrick (multi-layered raster) in `Anguilla_grids`.

```
data(Anguilla_grids)
plot(Anguilla_grids)
```



There is (obviously) no grid for fishing method. We create a data.frame with a constant value (of class 'factor') and pass that on to the predict function.

```
Method <- factor('electric', levels = levels(Anguilla_train$Method))
add <- data.frame(Method)
p <- predict(Anguilla_grids, angaus.tc5.lr005, const=add,
       n.trees=angaus.tc5.lr005$gbm.call$best.trees, type="response")
p <- mask(p, raster(Anguilla_grids, 1))
plot(p, main='Angaus - BRT prediction')
```



9.11 Further reading

Elith, J., Leathwick, J.R., and Hastie, T. (2008). Boosted regression trees - a new technique for modelling ecological data. Journal of Animal Ecology