Spatial Distribution Models

Robert J. Hijmans and Jane Elith

Nov 30, 2023

CONTENTS

Robert J. Hijmans and Jane Elith

[under development]

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 `terra` package.

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 `predicts` and the `terra` 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("terra", "remotes", "predicts", "geodata"))
```

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).

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 `predicts` 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 the `predicts` package is installed.

```
library(terra)
## terra 1.7.62
library(predicts)
filename <- file.path(system.file(package="predicts"), "ex/bradypus.csv")
# this is the file we will use:
basename(filename)
## [1] "bradypus.csv"
```

Now read the file and inspect the values.

```
bradypus <- read.csv(filename)
# 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
## 6 Bradypus variegatus -64.4167 -16.0000
# we only need columns 2 and 3:
bradypus <- bradypus[,2:3]
```

(continues on next page)

(continued from previous page)

```
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 from Excel files with the `readxl` package. 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 `geodata` package there is a function `sp_occurrence` that you can use for this. The data used below were downloaded, and saved to a permanent data set for use in this chapter, using the `sp_occurrence` function like this:

```
acaule <- geodata::sp_occurrence("solanum", "acaule*", geo=FALSE)
## Loading required namespace: jsonlite
## 7238 records found
## 0-300-600-900-1200-1500-1800-2100-2400-2700-3000-3300-3600-3900-4200
```

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 occurrence 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,

```
# load the saved S. acaule data
acfile <- file.path(system.file(package="predicts"), "ex/acaule.csv")
acaule <- read.csv(acfile)

# 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"
```

(continues on next page)

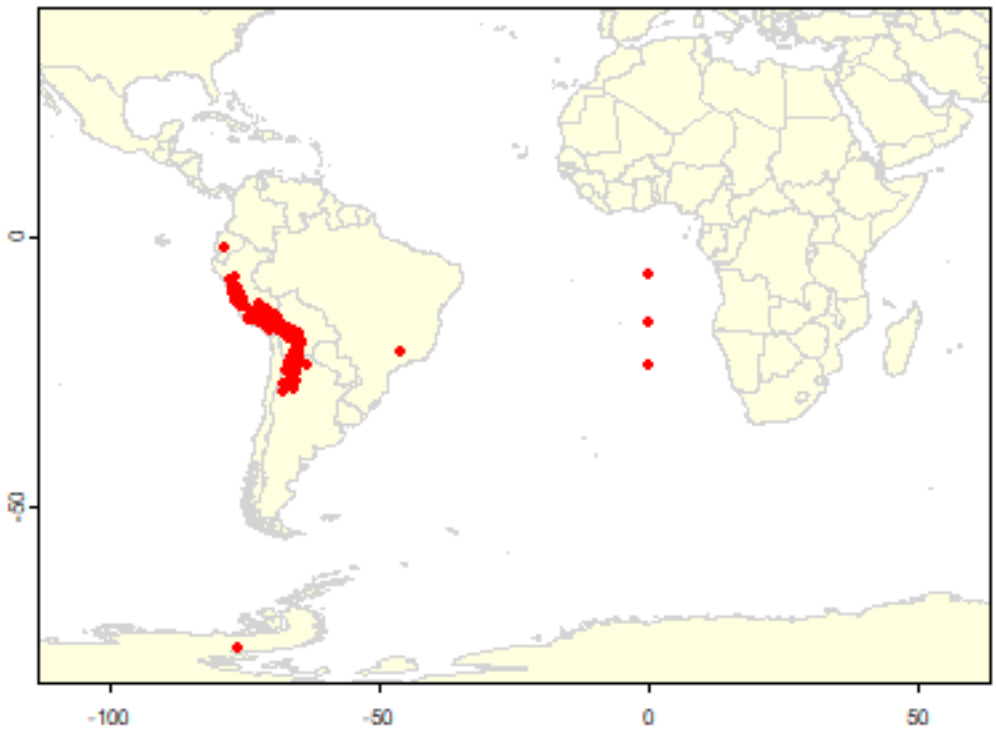
(continued from previous page)

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

# show some values
acgeo[1:5, 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>
## 5 Solanum acaule f. acaule          <NA>   Bolivia  <NA>           <NA>
##        lat      lon coordUncertaintyM  alt
## 1 -21.9000 -66.1000                NA   NA
## 2 -13.5000 -71.0000                NA 4500
## 3 -22.2666 -65.1333                NA 3800
## 4 -18.6333 -66.9500                NA 3700
## 5 -18.8000 -65.8833                NA 4080
```

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(geodata)
wrld <- world(path=".")
plot(wrld, xlim=c(-110,60), ylim=c(-80,40), col="light yellow", border="light gray")
# add the points
points(acgeo$lon, acgeo$lat, col='red', pch=20)
```



The `wrld` 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 `gadm` function of the `geodata` package. You can also read your own shapefile data using the `vect` function in the `terra` 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:

```
acgeo[c(303,885),1:10]
##                          species     continent country   adm1    adm2
## 328         solanum acaule acaule          <NA> Bolivia   <NA>    <NA>
## 1169 Solanum acaule subsp. acaule South America Bolivia La Paz Pacajes
##                                locality       lat     lon coordUncertaintyM
## 328                               <NA> -17.08300 -68.417                NA
## 1169 Pacajes. Caquiaviri, Kalla Centro. -17.01667 -68.600                NA
##       alt
## 328    NA
## 1169 3950
```

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$lon < -60 & acg$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).

We first make a `SpatVector`

```
library(terra)
acv <- vect(acg, geom=c("lon", "lat"), crs="+proj=longlat +datum=WGS84")
class(acv)
## [1] "SpatVector"
## attr(,"package")
## [1] "terra"
```

We can now use do a spatial query of the polygons in `wrld`

```
ovr <- extract(acv, wrld)
```

Object 'ovr' has, for each point, the matching record from `wrld`. We need the variable 'NAME_0' in the data.frame of wrld_simpl

```
head(ovr)
##   id.y species continent country adm1 adm2 locality coordUncertaintyM alt
## 1    1    <NA>      <NA>    <NA> <NA> <NA>     <NA>                NA  NA
## 2    2    <NA>      <NA>    <NA> <NA> <NA>     <NA>                NA  NA
## 3    3    <NA>      <NA>    <NA> <NA> <NA>     <NA>                NA  NA
## 4    4    <NA>      <NA>    <NA> <NA> <NA>     <NA>                NA  NA
## 5    5    <NA>      <NA>    <NA> <NA> <NA>     <NA>                NA  NA
## 6    6    <NA>      <NA>    <NA> <NA> <NA>     <NA>                NA  NA
##   institution collection catalogNumber basisOfRecord collector
## 1        <NA>       <NA>          <NA>          <NA>      <NA>
## 2        <NA>       <NA>          <NA>          <NA>      <NA>
## 3        <NA>       <NA>          <NA>          <NA>      <NA>
## 4        <NA>       <NA>          <NA>          <NA>      <NA>
## 5        <NA>       <NA>          <NA>          <NA>      <NA>
## 6        <NA>       <NA>          <NA>          <NA>      <NA>
##   earliestDateCollected latestDateCollected gbifNotes downloadDate
## 1                  <NA>                <NA>      <NA>         <NA>
## 2                  <NA>                <NA>      <NA>         <NA>
## 3                  <NA>                <NA>      <NA>         <NA>
## 4                  <NA>                <NA>      <NA>         <NA>
## 5                  <NA>                <NA>      <NA>         <NA>
## 6                  <NA>                <NA>      <NA>         <NA>
##   maxElevationM minElevationM maxDepthM minDepthM ISO2 cloc
## 1            NA            NA        NA        NA <NA> <NA>
## 2            NA            NA        NA        NA <NA> <NA>
## 3            NA            NA        NA        NA <NA> <NA>
## 4            NA            NA        NA        NA <NA> <NA>
## 5            NA            NA        NA        NA <NA> <NA>
## 6            NA            NA        NA        NA <NA> <NA>
cntr <- ovr$NAME_0
```

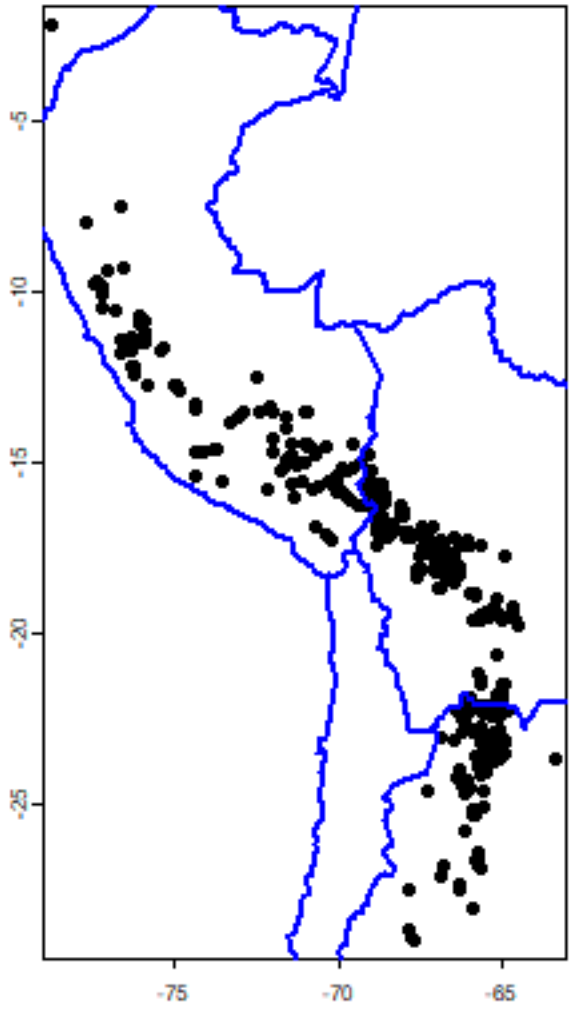
We should ask these two questions: (1) Which points (identified by their record numbers) do not match any country

(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 != acv$country)
# for the mismatches, bind the country names of the polygons and points
m <- cbind(cntr[j], acg$country[j])
colnames(m) <- c("polygons", "acaule")
m
##      polygons acaule
```

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(acv)
lines(wrld, col='blue', lwd=2)
```




```
points(acv[j, ], col='red', pch=20, cex=2)
```

The wrld 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 functions from the `predicts` package (e.g. `predicts::gadm(country='BOL', level=0)` to get the national borders of Bolivia; and `?predicts::world to get more precise boundaries for all countries).

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 predicts 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$cloc[4]
#b <- geocode(georef$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

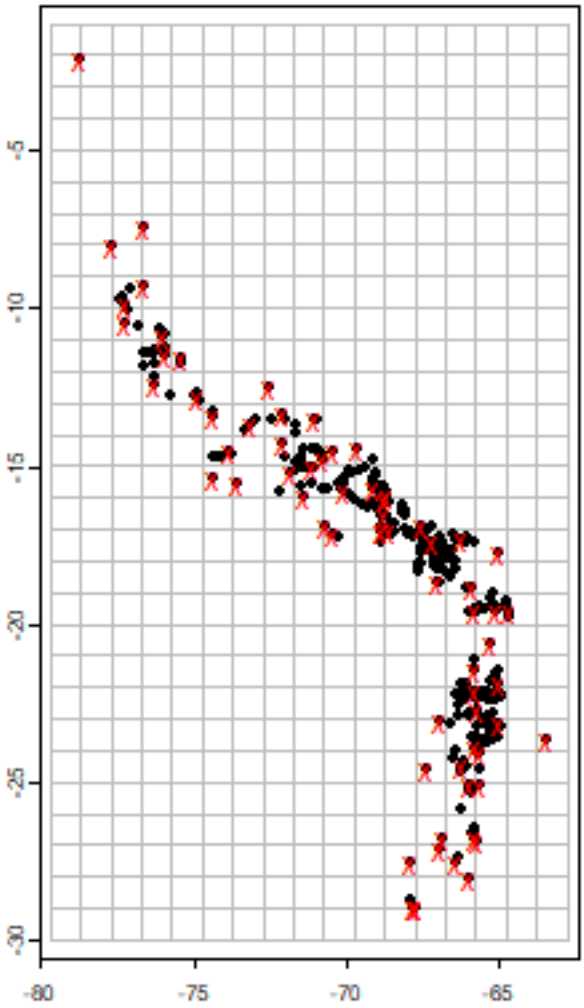
Sampling bias is frequently present in occurrence records (Hijmans *et al.*, 2001). One can attempt to remove some of the bias by subsampling records, and this is illustrated below. However, subsampling reduces the number of records, and it cannot correct the data for areas that have not been sampled 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 SpatRaster with the extent of acgeo
r <- rast(acv)
# set the resolution of the cells to (for example) 1 degree
res(r) <- 1

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

# sample:
set.seed(13)
acsel <- spatSample(acv, size=1, "random", strata=r)

# to illustrate the method and show the result
p <- as.polygons(r)
plot(p, border='gray')
points(acv)
# 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, you can use `as.data.frame(acsel)` and then `write.csv`. Or you can use `pack` and `saveRDS` so that we can use them later. We did that, and the saved file is available from the `predicts` package and can be read like this:

```
file <- paste(system.file(package="predicts"), '/ex/acaule.rds', sep='')
acsel <- readRDS(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 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 `terra` 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/latitude coordinate reference system, the `spatSample` method adjusts the probability of selecting a cells according to cell area, which varies by latitude.

```
library(predicts)
## Loading required package: terra
## terra 1.7.62
# get the predictors filename
f1 <- system.file("ex/bio.tif", package="predicts")
f2 <- system.file("ex/biome.tif", package="predicts")

r <- rast(c(f1, f2))
```

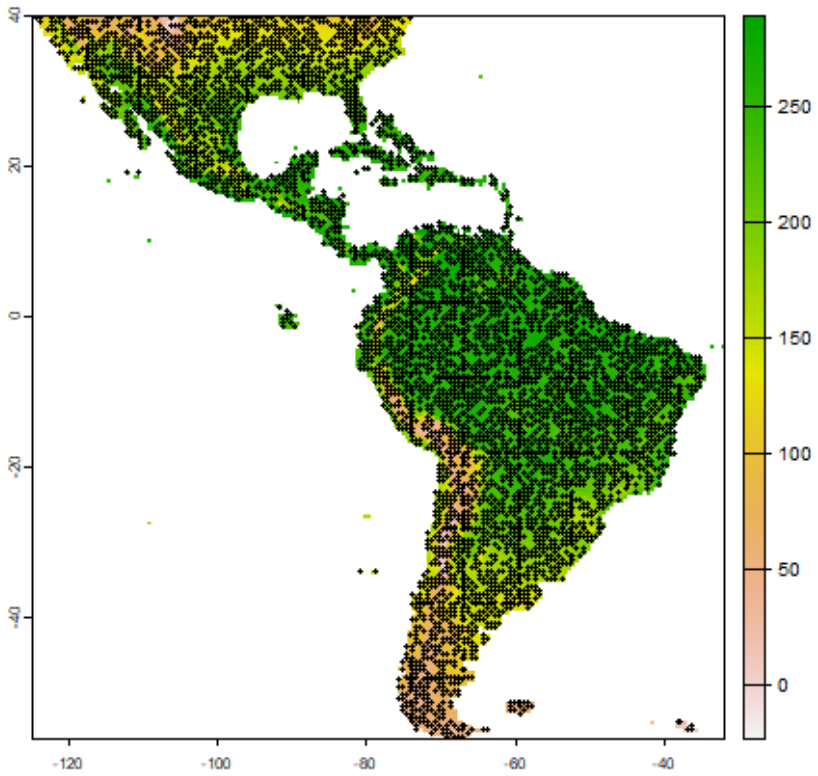
(continues on next page)

(continued from previous page)

```
# select 5000 random points
# set seed to assure that the examples will always
# have the same random sample.
set.seed(1963)
bg <- spatSample(r, 5000, "random", na.rm=TRUE, as.points=TRUE)
```

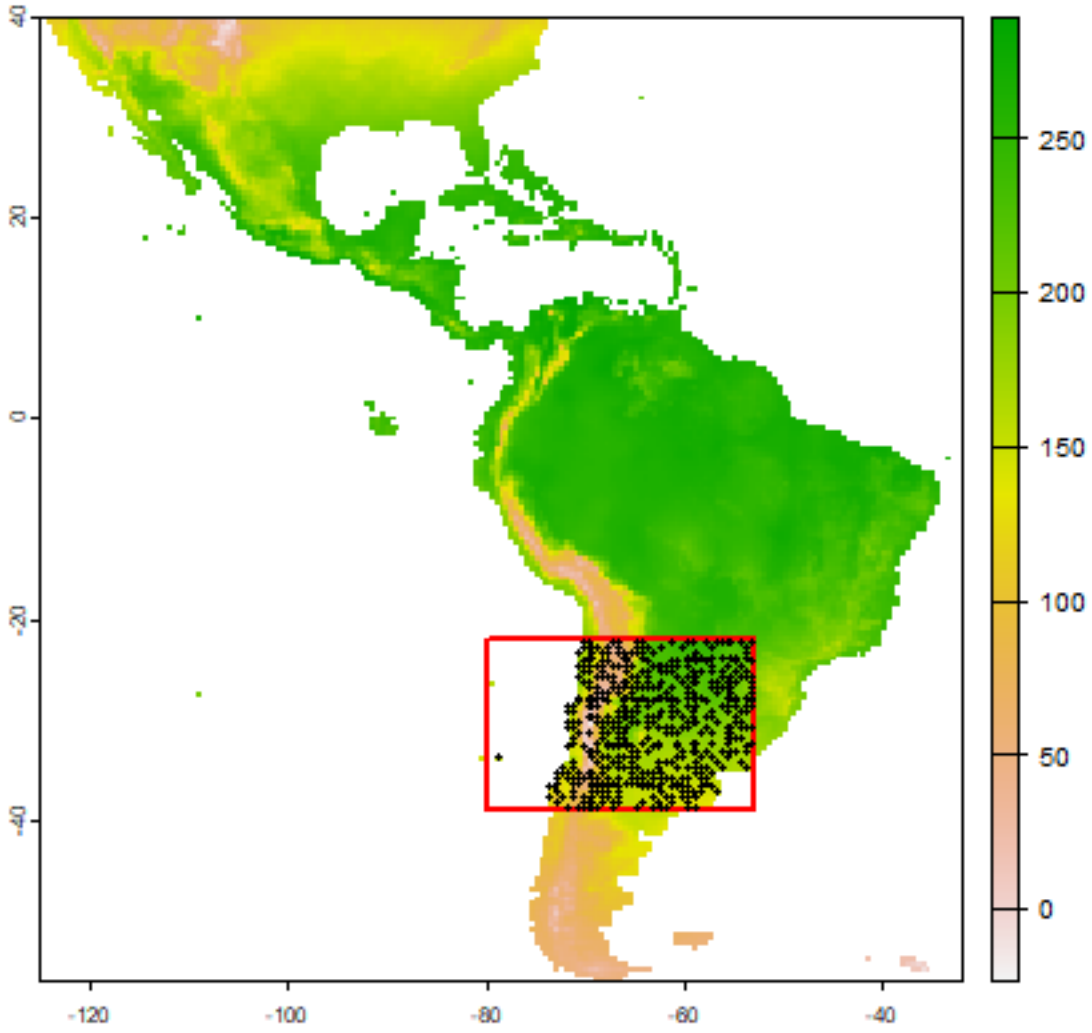
And inspect the results by plotting

```
plot(r, 1)
points(bg, cex=0.5)
```



Now we repeat the sampling, but limit the area of sampling using a spatial extent

```
e <- ext(-80, -53, -39, -22)
bg2 <- spatSample(r, 500, "random", na.rm=TRUE, as.points=TRUE, ext=e)
plot(r, 1)
lines(e, col="red", lwd=2)
points(bg2, cex=0.5)
```



There are several approaches one could use to sample 'pseudo-absence' points, i.e., points from a 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.

```
acfile <- file.path(system.file(package="predicts"), "ex/acaule.rds")
ac <- readRDS(acfile)
```

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 <- buffer(ac, 50000)
pol <- aggregate(x)
```

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

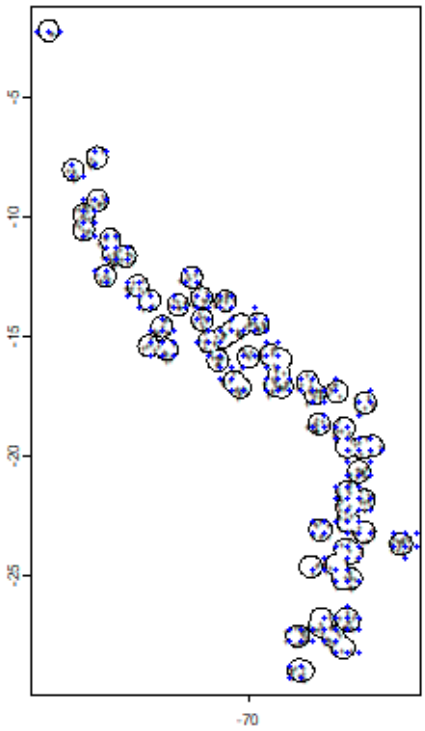
```
# sample randomly from all circles
set.seed(999)
samp1 <- spatSample(pol, 500, "random")
```

We do not what to include the same raster cells multiple times

```
pcells <- cells(r, samp1)
# remote duplicates
pcells <- unique(pcells[,2])
# back to coordinates
xy <- xyFromCell(r, pcells)
```

Plot to inspect the results:

```
plot(pol, axes=TRUE)
points(samp1, pch="+", cex=.5)
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. We could choose to select only those cells that have their centers within the circles, using the `intersect` method.

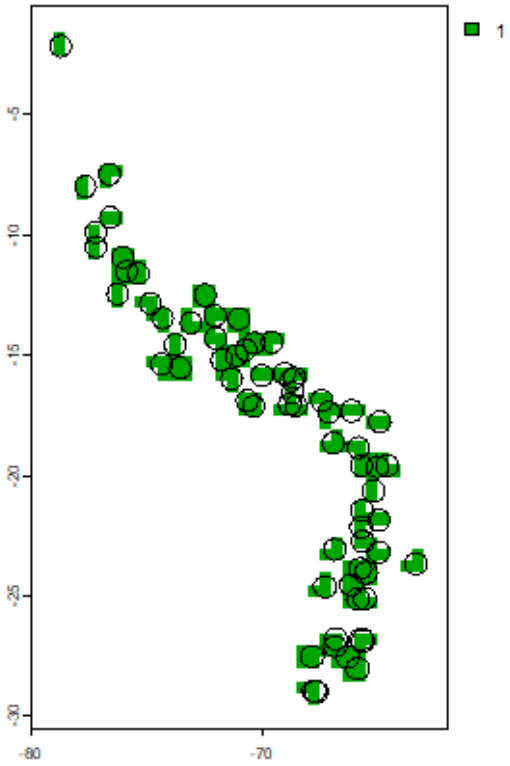
```
spxy <- vect(xy, crs="+proj=longlat +datum=WGS84")
xyInside <- intersect(spxy, x)
```

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

```
# make a new, empty, smaller raster
m <- crop(rast(r[[1]]), ext(x)+1)
# extract cell numbers for the circles
v <- cells(m, x)

# get unique cell numbers from which you could sample
v <- unique(v[,"cell"])
head(v)
## [1] 1505 1541  918  919  954  955

# to display the results
m[v] <- 1
plot(m)
lines(x)
```



CHAPTER

FOUR

ENVIRONMENTAL DATA

4.1 Raster data

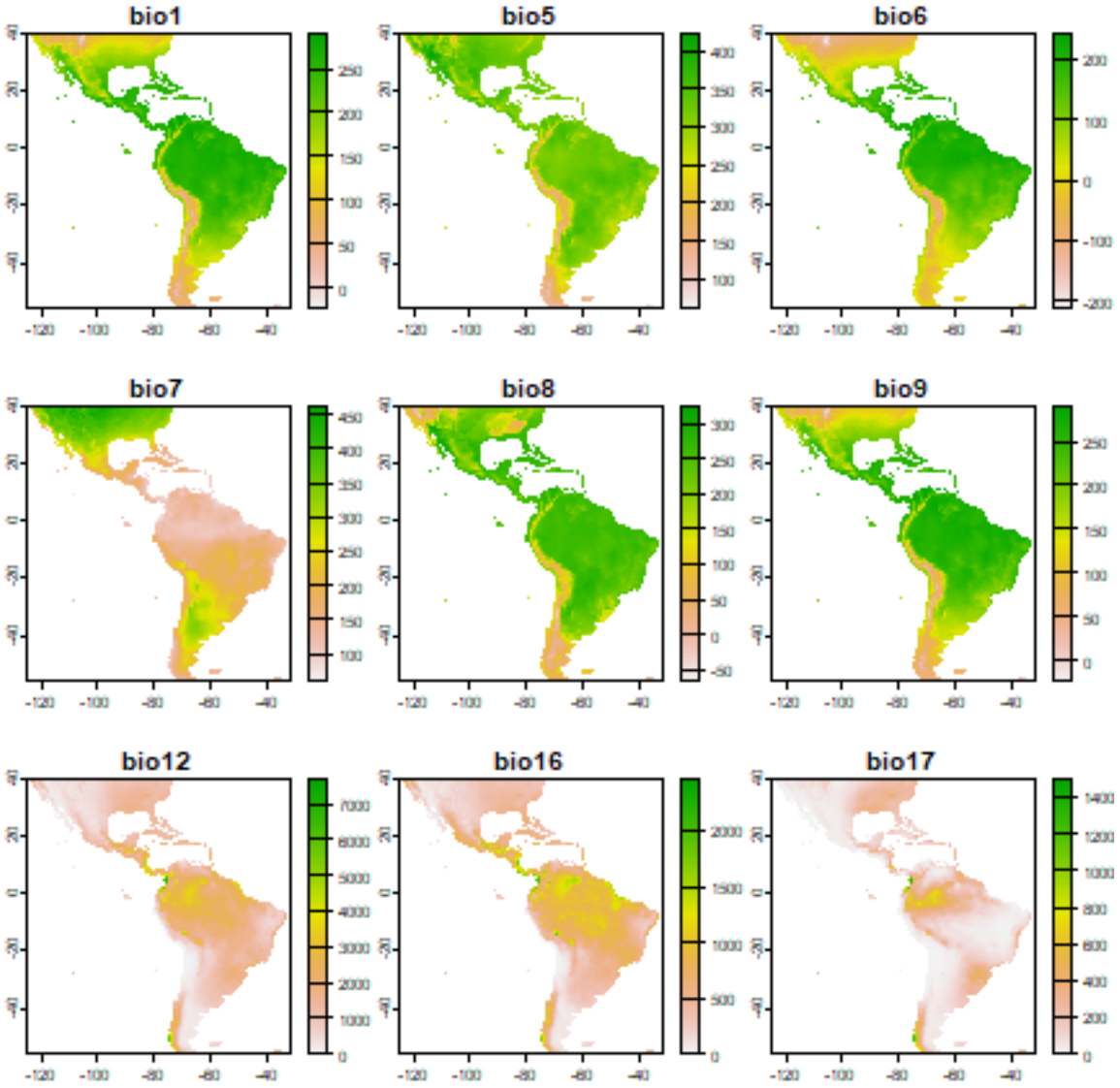
In species distribution modeling, predictor variables are typically organized as raster (grid) type files. The predictors should be layers in a `SpatRaster` representing variables of interest such as climate, soil, terrain, vegetation, or land use. These data are typically stored in files in some kind of geospatial format. Almost all relevant formats can be used (including geoTiff, netCDF, and Arc-ASCII). 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 `project` from the `terra` 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 `SpatRaster`, which has many layers (see the Spatial Data tutorial for more info.

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

First get the folder with our files. Here we use a file that ships with R.

```
library(predicts)
## Loading required package: terra
## terra 1.7.62
f <- system.file("ex/bio.tif", package="predicts")
predictors <- rast(f)
predictors
## class       : SpatRaster
## dimensions  : 192, 186, 9  (nrow, ncol, nlyr)
## resolution  : 0.5, 0.5  (x, y)
## extent      : -125, -32, -56, 40  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : bio.tif
## names       : bio1, bio5, bio6, bio7, bio8, bio9, ...
## min values  :  -23,   61, -212,   60,  -66,  -23, ...
## max values  :  289,  422,  242,  461,  323,  289, ...
names(predictors)
## [1] "bio1"  "bio5"  "bio6"  "bio7"  "bio8"  "bio9"  "bio12" "bio16" "bio17"
plot(predictors)
```

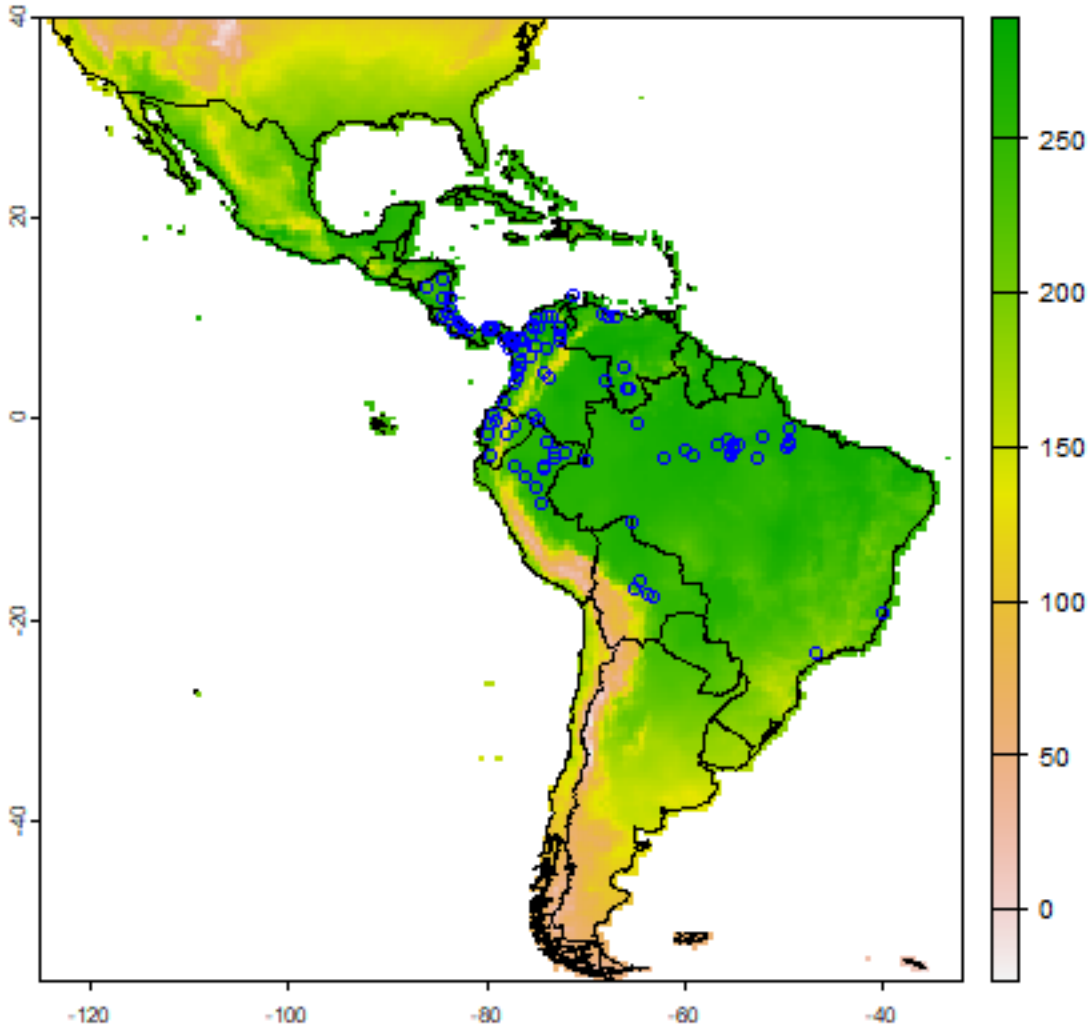


We can also make a plot of a single layer in a SpatRaster, and plot some additional data on top of it. First get the world boundaries and the *Bradypus* data:

```
library(geodata)
wrld <- world(path=".")
file <- paste(system.file(package="predicts"), "/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 SpatRaster
plot(predictors, 1)
lines(wrld)
points(bradypus, col='blue')
```



The example above uses data representing 'bioclimatic variables' from the WorldClim database (Hijmans *et al.*, 2004). You can download more recent versions of WorldClim with the `geodata` package.

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.

```
presvals <- extract(predictors, bradypus)
# remove the ID variable
presvals <- presvals[,-1]
# setting random seed to always create the same
# random set of points for this example
set.seed(0)
backgr <- spatSample(predictors, 500, "random", as.points=TRUE, na.rm=TRUE)
absvals <- values(backgr)

pb <- c(rep(1, nrow(presvals)), rep(0, nrow(absvals)))
sdmdata <- data.frame(cbind(pb, rbind(presvals, absvals)))
head(sdmdata)
##   pb bio1 bio5 bio6 bio7 bio8 bio9 bio12 bio16 bio17
## 1  1  263  338  191  147  261  263  1639   724    62
## 2  1  263  338  191  147  261  263  1639   724    62
## 3  1  253  329  150  179  271  253  3624  1547   373
## 4  1  243  318  150  168  264  243  1693   775   186
## 5  1  243  318  150  168  264  243  1693   775   186
## 6  1  252  326  154  172  270  252  2501  1081   280
tail(sdmdata)
##     pb bio1 bio5 bio6 bio7 bio8 bio9 bio12 bio16 bio17
## 611  0  263  324  207  117  260  263  2512   984   206
## 612  0  251  326  154  172  264  251  2053   886   194
## 613  0  235  311  141  170  244  235  1584   734    65
## 614  0  179  265   91  174  188  179  1633   727   125
## 615  0  254  325  182  143  257  254  1384   492   233
## 616  0  138  344  -70  414  229  138   513   224    42
summary(sdmdata)
##        pb              bio1            bio5            bio6
##  Min.   :0.0000   Min.   : -8.0   Min.   : 90.0   Min.   :-182.00
##  1st Qu.:0.0000   1st Qu.:169.0   1st Qu.:299.0   1st Qu.:  33.25
##  Median :0.0000   Median :238.5   Median :318.0   Median : 150.00
##  Mean   :0.1883   Mean   :209.8   Mean   :305.1   Mean   : 114.67
##  3rd Qu.:0.0000   3rd Qu.:259.0   3rd Qu.:331.2   3rd Qu.: 198.00
##  Max.   :1.0000   Max.   :286.0   Max.   :413.0   Max.   : 236.00
##       bio7            bio8            bio9           bio12
##  Min.   : 64.0   Min.   :-38.0   Min.   : -8.0   Min.   :   1.0
##  1st Qu.:119.8   1st Qu.:212.0   1st Qu.:169.0   1st Qu.: 757.2
##  Median :163.0   Median :249.5   Median :238.5   Median :1387.5
##  Mean   :190.4   Mean   :221.8   Mean   :209.8   Mean   :1532.2
##  3rd Qu.:247.2   3rd Qu.:261.0   3rd Qu.:259.0   3rd Qu.:2192.8
##  Max.   :444.0   Max.   :316.0   Max.   :286.0   Max.   :7682.0
##      bio16           bio17
##  Min.   :   1.0   Min.   :   0.00
```

(continues on next page)

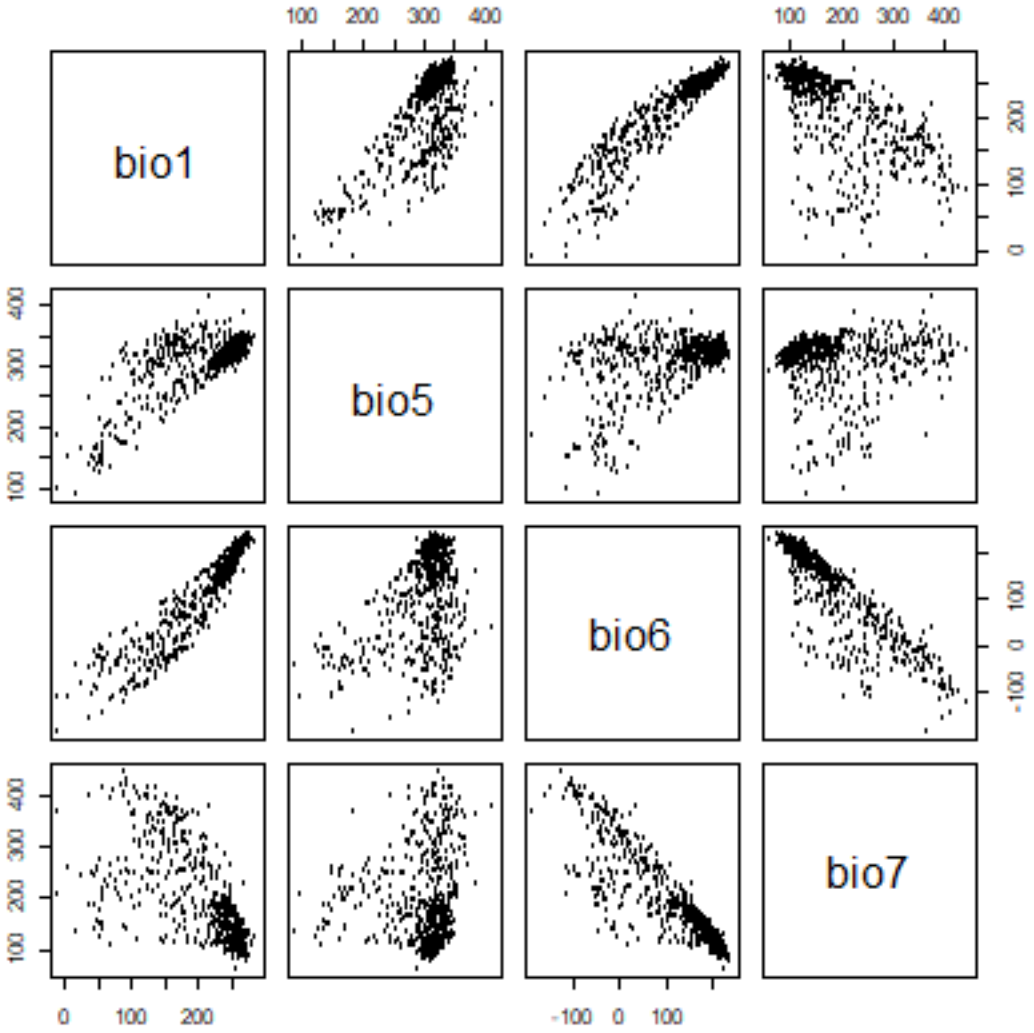
(continued from previous page)

```
##  1st Qu.: 311.8   1st Qu.:  38.75
##  Median : 596.0   Median : 103.50
##  Mean   : 624.0   Mean   : 157.59
##  3rd Qu.: 895.2   3rd Qu.: 219.25
##  Max.   :2458.0   Max.   :1496.00
```

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).

```
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)
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.265e-02  9.729e-02   0.952 0.341341
## bio1         1.351e-03  3.890e-04   3.472 0.000554 ***
## bio5        -1.337e-03  4.495e-04  -2.973 0.003060 **
## bio12        1.437e-04  1.662e-05   8.648  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1158131)
##
##     Null deviance: 94.156  on 615  degrees of freedom
## Residual deviance: 70.878  on 612  degrees of freedom
## AIC: 426.16
##
```

(continues on next page)

(continued from previous page)

```
## Number of Fisher Scoring iterations: 2

m2 = glm(pb ~ ., data=sdmdata)
m2
##
## Call:  glm(formula = pb ~ ., data = sdmdata)
##
## Coefficients:
## (Intercept)         bio1         bio5         bio6         bio7         bio8
##   0.1844517   -0.0047289    0.0193746   -0.0155329   -0.0184310    0.0008068
##        bio9        bio12        bio16        bio17
##          NA    0.0004841   -0.0006574   -0.0008551
##
## Degrees of Freedom: 615 Total (i.e. Null);  607 Residual
## Null Deviance:       94.16
## Residual Deviance: 68.18     AIC: 412.2
```

Models that are implemented in the `predicts` package do not use a formula (and most models only take presence points). The `envelope` (a.k.a. "Bioclim") is an example. It only uses presence data, so we use 'presvals' instead of 'sdmdata'.

```
library(predicts)
## Loading required package: terra
## terra 1.7.62
bc <- envelope(presvals[,c("bio1", "bio5", "bio12")])
bc
## Formal class 'envelope_model' [package "predicts"] with 7 slots
##   ..@ min       : Named num [1:3] 150 206 372
##   .. ..- attr(*, "names")= chr [1:3] "bio1" "bio5" "bio12"
##   ..@ max       : Named num [1:3] 282 351 7682
##   .. ..- attr(*, "names")= chr [1:3] "bio1" "bio5" "bio12"
##   ..@ factor    : list()
##   ..@ names     : chr [1:3] "bio1" "bio5" "bio12"
##   ..@ presence  :'data.frame':   116 obs. of  3 variables:
##   .. ..$ bio1 : num [1:116] 263 263 253 243 243 252 240 275 271 274 ...
##   .. ..$ bio5 : num [1:116] 338 338 329 318 318 326 317 335 327 329 ...
##   .. ..$ bio12: num [1:116] 1639 1639 3624 1693 1693 ...
##   ..@ absence   :'data.frame':   0 obs. of  0 variables
## Formal class 'data.frame' [package "methods"] with 4 slots
##   .. .. ..@ .Data    : list()
##   .. .. ..@ names    : chr(0)
##   .. .. ..@ row.names: int(0)
##   .. .. ..@ .S3Class : chr "data.frame"
##   ..@ hasabsence: logi FALSE
```

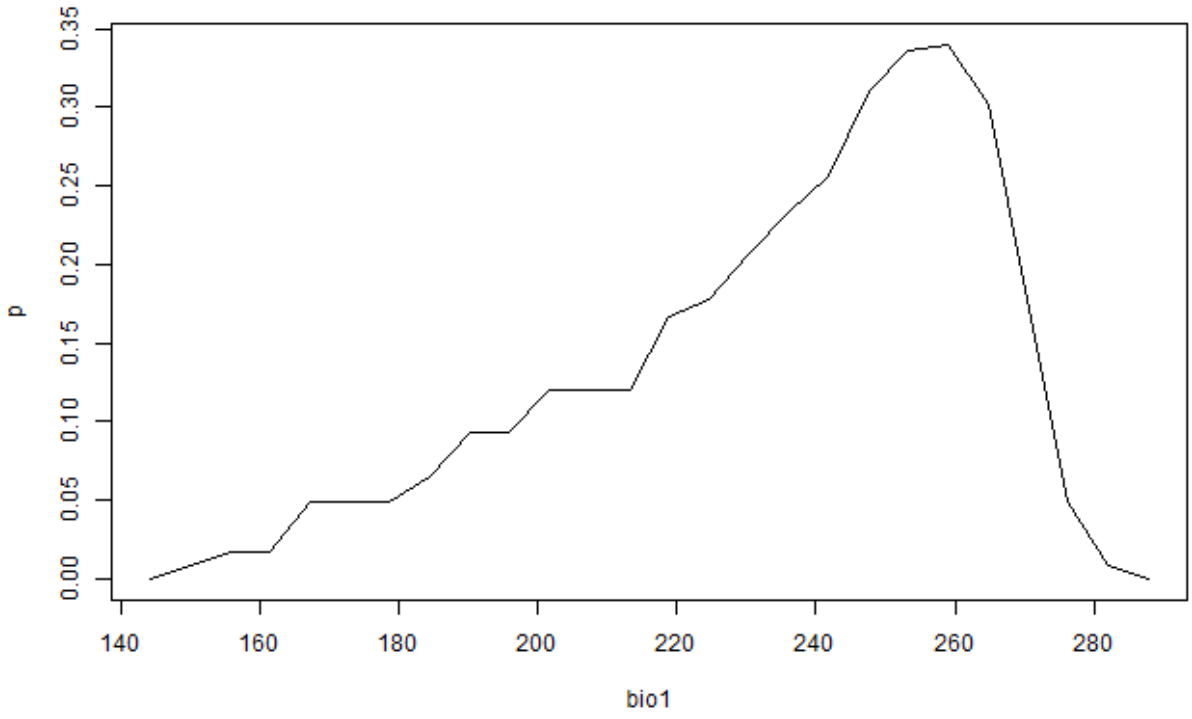
5.2 Model prediction

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 envelope 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)
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
## 0.1527031 0.3714683 0.2194479
```

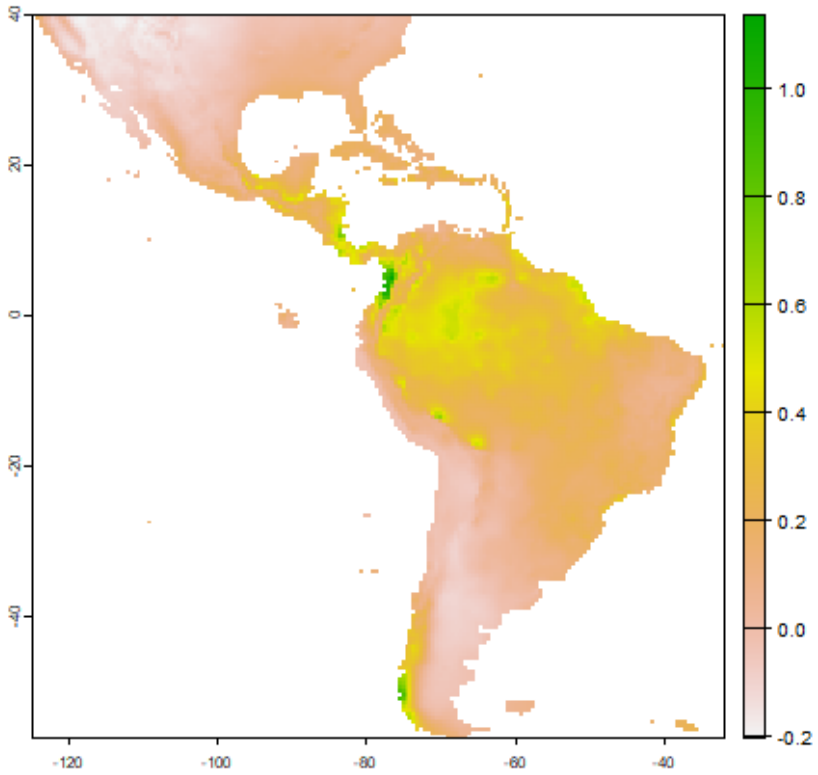
Making such predictions for a few environments can be very useful to explore and understand model predictions. For example it used in the `response` function that creates response plots for each variable, with the other variables at their median value.

```
pr <- partialResponse(bc, presvals, "bio1")
plot(pr, type="l")
```



In most cases, howver, the purpose of 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 in the SpatRaster object.

```
library(predicts)
f <- system.file("ex/bio.tif", package="predicts")
predictors <- rast(f)
names(predictors)
## [1] "bio1"  "bio5"  "bio6"  "bio7"  "bio8"  "bio9"  "bio12" "bio16" "bio17"
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)

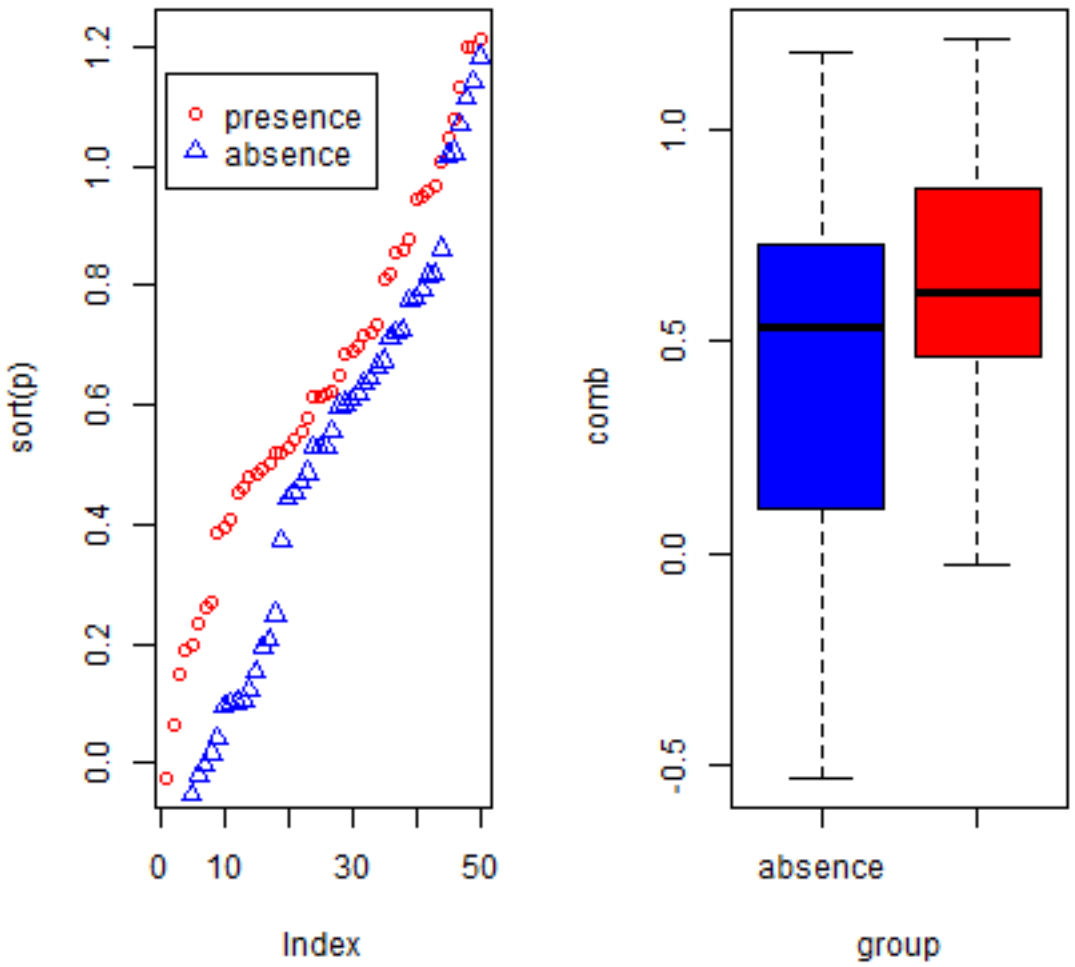
Most modelers rely on cross-validation. This consists of creating a model with one 'training' data set, and testing it with another data set of known occurrences. Typically, training and testing data are created through random sampling (without replacement) from a single data set. Only in a few cases, 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'))
```

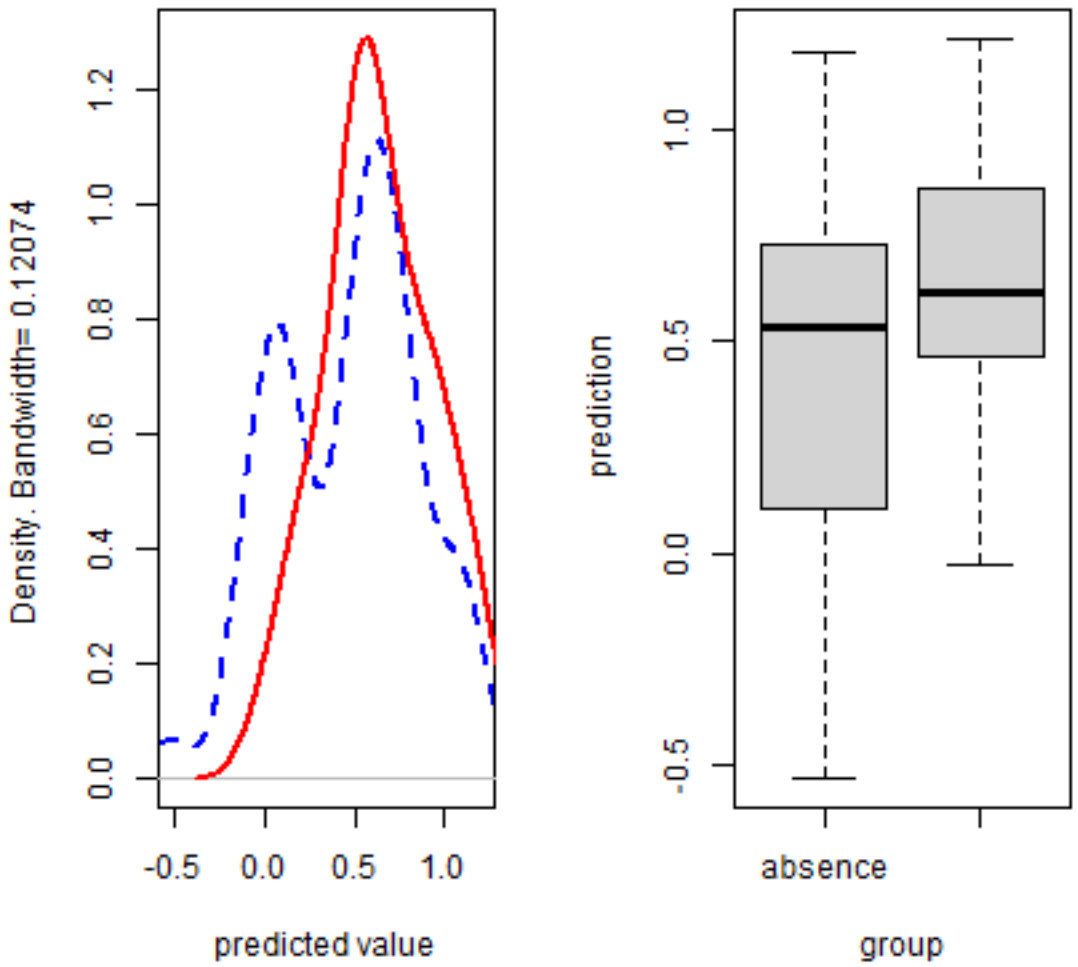


We created two variables with random normally distributed values, but with different mean and standard deviation. The two variables clearly have different distributions, and the values for 'presence' tend to be higher than for 'absence'. Here is how you can compute the correlation coefficient and the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

```
group = c(rep(1, length(p)), rep(0, length(a)))
cor.test(comb, group)$estimate
##      cor
## 0.238617
mv <- wilcox.test(p,a)
auc <- as.numeric(mv$statistic) / (length(p) * length(a))
auc
## [1] 0.6172
```

Below we show how you can compute these, and other statistics more conveniently, with the `evaluate` function in the predicts package. See `?pa_evaluate` for info on additional evaluation measures that are available.

```
library(predicts)
e <- pa_evaluate(p=p, a=a)
class(e)
## [1] "paModelEvaluation"
## attr(,"package")
## [1] "predicts"
e
## @stats
##   np na prevalence   auc   cor  pcor ODP
## 1 50 50        0.5 0.617 0.239 0.017 0.5
##
## @thresholds
##   max_kappa max_spec_sens no_omission equal_prevalence equal_sens_spec
## 1      0.15          0.15      -0.026            0.503           0.597
##
## @tr_stats
##     treshold kappa  CCR  TPR  TNR  FPR  FNR  PPP  NPP  MCR  OR
## 1      -0.53     0  0.5    1    0    1    0  0.5  NaN  0.5 NaN
## 2      -0.17  0.02 0.51    1 0.02 0.98    0 0.51    1 0.49 Inf
## 3      -0.11  0.04 0.52    1 0.04 0.96    0 0.51    1 0.48 Inf
## 4        ...   ...  ...  ...  ...  ...  ...  ...  ...  ... ...
## 100     1.21  0.02 0.51 0.02    1    0 0.98    1 0.51 0.49 Inf
## 101     1.21  0.02 0.51 0.02    1    0 0.98    1 0.51 0.49 Inf
## 102     1.21     0  0.5    0    1    0    1  NaN  0.5  0.5 NaN
par(mfrow=c(1, 2))
plot(e, "density")
plot(e, "boxplot", col=c('blue', 'red'))
```



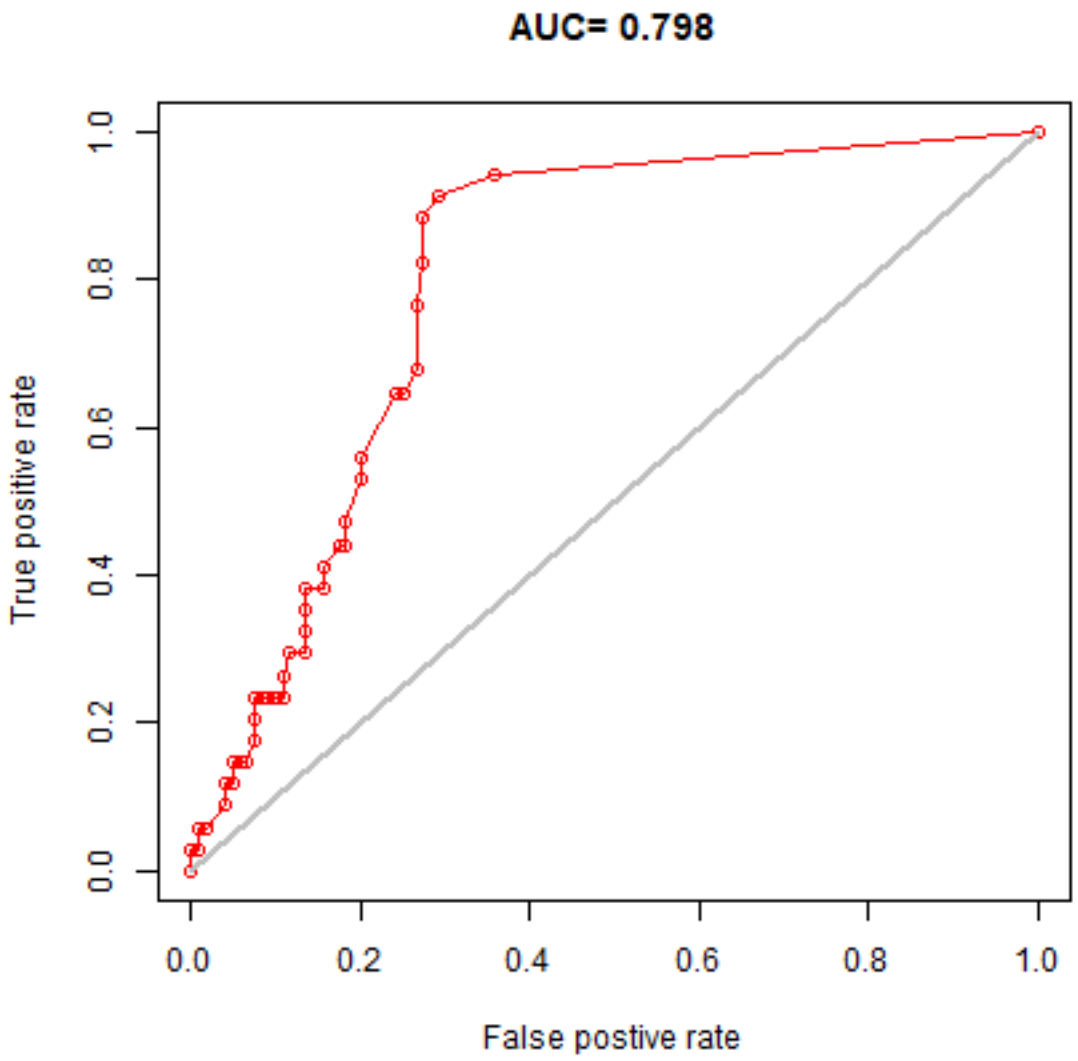
Back to some real data, presence-only in this case. We'll divide the data in two random sets, one for training a Bioclim model, and one for evaluating the model.

```
samp <- sample(nrow(sdmdata), round(0.75 * nrow(sdmdata)))
traindata <- sdmdata[samp,]
traindata <- traindata[traindata[,1] == 1, 2:9]
testdata <- sdmdata[-samp,]
bc <- envelope(traindata)
e <- pa_evaluate(testdata[testdata==1,], testdata[testdata==0,], bc)
e
## @stats
##   np  na prevalence   auc  cor pcor   ODP
## 1 34 120      0.221 0.798 0.33    0 0.779
##
## @thresholds
##   max_kappa max_spec_sens no_omission equal_prevalence equal_sens_spec
## 1     0.049         0.036           0            0.219           0.085
```

(continues on next page)

(continued from previous page)

```
##
## @tr_stats
##    treshold kappa  CCR  TPR  TNR  FPR  FNR  PPP  NPP  MCR    OR
## 1         0     0 0.22    1    0    1    0 0.22  NaN 0.78   NaN
## 2      0.02  0.41 0.71 0.94 0.64 0.36 0.06 0.43 0.97 0.29 28.65
## 3      0.04  0.46 0.75 0.91 0.71 0.29 0.09 0.47 0.97 0.25  25.1
## 4       ...   ...  ...  ...  ...  ...  ...  ...  ...  ...   ...
## 39      0.8  0.05 0.79 0.03    1    0 0.97    1 0.78 0.21   Inf
## 40      0.8     0 0.78    0    1    0    1  NaN 0.78 0.22   NaN
## 41      0.8     0 0.78    0    1    0    1  NaN 0.78 0.22   NaN
plot(e, "ROC")
```



In real projects, you would want to use `k-fold` data partitioning instead of a single random sample. The `predicts::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 <- folds(pres, k)
group[1:10]
##  [1] 4 3 4 3 4 1 2 2 1 1
unique(group)
## [1] 4 3 1 2 5
table(group)
## group
##  1  2  3  4  5
## 23 23 24 23 23
```

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 <- envelope(train)
    p <- predict(bc, test)
    a <- predict(bc, back)
    e[[i]] <- pa_evaluate(p, a)
}
```

We can extract several things from list `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).

```
stats <- lapply(e, function(x){x@stats})
stats <- do.call(rbind, stats)
colMeans(stats)
##           np           na   prevalence          auc          cor         pcor
## 2.320000e+01 5.000000e+02 4.434195e-02 7.923536e-01 1.892941e-01 3.916510e-04
##          ODP
## 9.556581e-01
```

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".

```
fsp <- system.file("/ex/bradypus.csv", package="predicts")
bradypus <- read.csv(fsp)
```

(continues on next page)

(continued from previous page)

```
bradypus <- bradypus[,-1]
presvals <- extract(predictors, bradypus)
set.seed(0)
backgr <- spatSample(predictors, 500, na.rm=TRUE, xy=TRUE, values=FALSE)

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, ]
```

```
library(dismo)
## Loading required package: raster
## Loading required package: sp
##
## Attaching package: 'sp'
## The following object is masked from 'package:predicts':
##
##     geometry
##
## Attaching package: 'dismo'
## The following objects are masked from 'package:predicts':
##
##     mess, threshold
sb <- dismo::ssb(pres_test, back_test, pres_train)
sb[,1] / sb[,2]
##          p
## 0.06266938
```

`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 <- pwd_sample(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 <- dismo::ssb(pres_test_pwd, back_test_pwd, pres_train)
sb2[1]/ sb2[2]
## [1] 1.000111
```

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

```
bc <- envelope(predictors, pres_train)
ptst <- predict(bc, extract(predictors, pres_test))
btst <- predict(bc, extract(predictors, back_test))
pa_evaluate(ptst, btst)@stats
##   np  na prevalence       auc       cor         pcor       ODP
## 1 29 125  0.1883117 0.8097931 0.3608806 4.269173e-06 0.8116883
```

(continues on next page)

(continued from previous page)

```
pwdptst <- predict(bc, extract(predictors, pres_test_pwd))
pwdbtst <- predict(bc, extract(predictors, back_test_pwd))
pa_evaluate(pwdptst, pwdbtst)@stats
##   np na prevalence       auc       cor      pcor ODP
## 1 11 11        0.5 0.6280992 0.1496341 0.5062836 0.5
```

CHAPTER

SIX

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.