Species distribution modeling with R

Robert J. Hijmans and Jane Elith

November 18, 2013

Chapter 1

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 assumption of SDM is that you can predict the entire, or potential, spatial distribution of a phenomenon, by relating sites of known occurrence (and perhaps non-occurrence) with predictor variables known for these sites and for all other sites. The 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 predicting likelihood of presence, or another measure such as abundance of the species. (4) The model is used to predict the likelihood of presence across an area of interest (and perhaps for a future or past climate).

In this text we assume that you are familiar with most of the concepts in SDM. If in doubt, you could consult, for example, Richard Pearson's introduction to the subject: `http://biodiversityinformatics.amnh.org/index.php?section_id=111`, the book by Janet Franklin (2009), or 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 (http://cran.r-project.org/) and any introduction to R txt. For the '`raster`' package you could consult its vignette (available at `http://cran.r-project.org/web/packages/raster/vignettes/Raster.pdf`). 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.

Part I

Data preparation

Chapter 2

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, Quantum GIS, `http://www.qgis.org/`, is a very useful program for interactive editing of point data sets.

2.1 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. The R commands used are in *italics* and preceded by a '>'. Comments are preceded by a hash (#). 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 <- paste(system.file(package="dismo"), "/ex/bradypus.csv", sep="")
> # this is the file we will use:
> file
```

```
[1] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bradypus.csv"
```

```
> # let's read it
> bradypus <- read.table(file,  header=TRUE,  sep=',')
> # let's inspect the values of the 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
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) (`http://www.gbif.org/`). 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*.

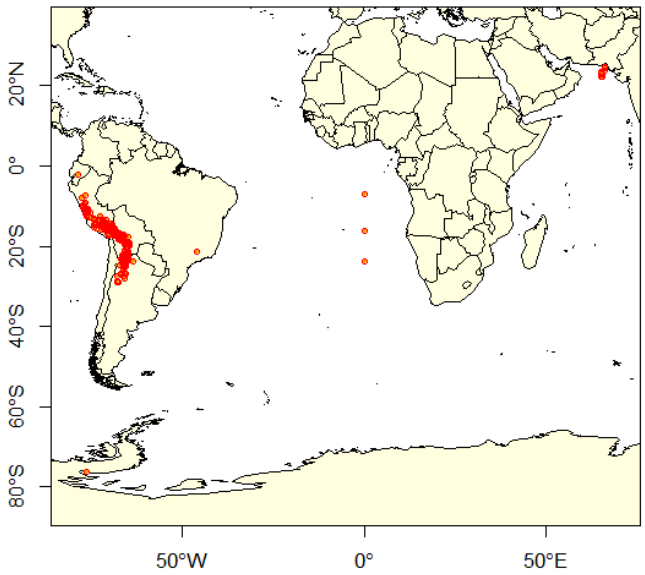
Many occurence records may not have geographic coordinates. In this case, out of the 1366 records that GBIF returned (January 2013), there were 1082 records with coordinates (this was 699 and 54 in March 2010, a tremendous improvement!)

```
> # 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"
 [3] "country"               "adm1"
 [5] "adm2"                  "locality"
 [7] "lat"                   "lon"
 [9] "coordUncertaintyM"     "alt"
[11] "institution"           "collection"
[13] "catalogNumber"         "basisOfRecord"
[15] "collector"             "earliestDateCollected"
[17] "latestDateCollected"   "gbifNotes"
[19] "downloadDate"          "maxElevationM"
[21] "minElevationM"         "maxDepthM"
[23] "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
1    Solanum acaule Bitter South America Argentina Jujuy
2    Solanum acaule Bitter South America      Peru Cusco
```

```
3 Solanum acaule f. acaule          <NA> Argentina  <NA>
4 Solanum acaule f. acaule          <NA>   Bolivia  <NA>
            adm2      lat      lon coordUncertaintyM  alt
1 Santa Catalina -21.9000 -66.1000              <NA>  NaN
2        Canchis -13.5000 -71.0000              <NA> 4500
3           <NA> -22.2666 -65.1333              <NA> 3800
4           <NA> -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()
> # plot 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.

2.2 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
303        solanum acaule acaule      <NA> Antarctica <NA>
885 solanum acaule acaule BITTER      <NA>       Peru <NA>
    adm2                     locality   lat   lon
303 <NA>                         <NA> -76.3 -76.3
885 <NA> Lima P. Huarochiri Pacomanta -76.3 -76.3
    coordUncertaintyM  alt
303              <NA>  NaN
885              <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
1159 Solanum acaule Bitter subsp. acaule      <NA>
1160 Solanum acaule Bitter subsp. acaule      <NA>
1161 Solanum acaule Bitter subsp. acaule      <NA>
1162 Solanum acaule Bitter subsp. acaule      <NA>
1163 Solanum acaule Bitter subsp. acaule      <NA>
1164 Solanum acaule Bitter subsp. acaule      <NA>
       country adm1 adm2
1159 Argentina <NA> <NA>
1160   Bolivia <NA> <NA>
1161      Peru <NA> <NA>
1162      Peru <NA> <NA>
1163 Argentina <NA> <NA>
1164   Bolivia <NA> <NA>
                                          locality
1159 between Quelbrada del Chorro and Laguna Colorada
1160                                          Llave
1161                   km 205 between Puno and Cuzco
1162                   km 205 between Puno and Cuzco
1163 between Quelbrada del Chorro and Laguna Colorada
1164                                          Llave
            lat lon coordUncertaintyM  alt institution
1159 -23.716667   0              <NA> 3400         IPK
1160 -16.083334   0              <NA> 3900         IPK
1161  -6.983333   0              <NA> 4250         IPK
1162  -6.983333   0              <NA> 4250         IPK
1163 -23.716667   0              <NA> 3400         IPK
1164 -16.083334   0              <NA> 3900         IPK
     collection catalogNumber
1159         GB     WKS 30027
1160         GB     WKS 30050
1161  WKS 30048        304709
1162         GB     WKS 30048
1163  WKS 30027        304688
1164  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 texitS. albicans or an anomalous hexaploid variety of texitS. acaule.

2.2.1 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
1162 Solanum acaule Bitter subsp. acaule      <NA>
1163 Solanum acaule Bitter subsp. acaule      <NA>
1164 Solanum acaule Bitter subsp. acaule      <NA>
       country adm1 adm2
1162      Peru <NA> <NA>
1163 Argentina <NA> <NA>
1164   Bolivia <NA> <NA>
                                            locality
1162                    km 205 between Puno and Cuzco
1163 between Quelbrada del Chorro and Laguna Colorada
1164                                          Llave
            lat lon coordUncertaintyM  alt institution
1162  -6.983333   0              <NA> 4250         IPK
1163 -23.716667   0              <NA> 3400         IPK
1164 -16.083334   0              <NA> 3900         IPK
     collection catalogNumber
1162         GB     WKS 30048
1163  WKS 30027        304688
1164  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, ]
```

2.3 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 `overlay` function, also from `sp`, to do a point-in-polygon query with the countries polygons.

```
> library(sp)
> # make a SpatialPointsDataFrame
> coordinates(acg) <- ~lon+lat
> projection(acg) <- CRS('+proj=lonlat')
> class(acg)

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

> # 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"
```

```
> ov <- overlay(acg, wrld_simpl)
> # ov has, for each point, the record number of wrld_simpl
> # we can use the record number to extract the country name
> # first find the variable name
> names(wrld_simpl)

 [1] "FIPS"      "ISO2"      "ISO3"      "UN"
 [5] "NAME"      "AREA"      "POP2005"   "REGION"
 [9] "SUBREGION" "LON"       "LAT"

> head(wrld_simpl)

    FIPS ISO2 ISO3 UN                NAME   AREA  POP2005
ATG   AC   AG  ATG 28 Antigua and Barbuda     44    83039
DZA   AG   DZ  DZA 12             Algeria 238174 32854159
AZE   AJ   AZ  AZE 31          Azerbaijan   8260  8352021
ALB   AL   AL  ALB  8             Albania   2740  3153731
ARM   AM   AM  ARM 51             Armenia   2820  3017661
AGO   AO   AO  AGO 24              Angola 124670 16095214
    REGION SUBREGION     LON     LAT
ATG     19        29 -61.783  17.078
DZA      2        15   2.632  28.163
AZE    142       145  47.395  40.430
ALB    150        39  20.068  41.143
ARM    142       145  44.563  40.534
AGO      2        17  17.544 -12.296

> # We need the variable 'NAME' in the data.frame of wrld_simpl
>
> cntr <- as.character(wrld_simpl$NAME[ov])
> # which points (identified by their record numbers) do not match
> # any country (i.e. are in an ocean)
> i <- which(is.na(cntr))
> i

integer(0)

> # there are none (because we already removed the points that mapped in the ocean)
>
> # which points have coordinates that are in a different country than
> #  listed in the 'country' field of the gbif record
> j <- which(cntr != acg$country)
> # for the mismatches, bind the country names of the polygons and points
> cbind(cntr, acg$country)[j,]

     cntr
[1,] "Bolivia" "Argentina"
```

```
[2,] "Peru"    "Bolivia"
[3,] "Peru"    "Bolivia"
[4,] "Peru"    "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 `overlay` function and the related function `over`. At first it may be confusing that it returns indices (row numbers). These indices, stored in variables `i` and `j` were used to get the relevant records. 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 `http://www.gadm.org/` 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).

2.4 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
606 solanum acaule acaule BITTER      <NA> Bolivia <NA>
607 solanum acaule acaule BITTER      <NA>    Peru <NA>
618 solanum acaule acaule BITTER      <NA>    Peru <NA>
    adm2
606 <NA>
607 <NA>
618 <NA>
                                                                locality
606 La Paz P. Franz Tamayo Viscachani 3 km from Huaylapuquio to Pelechuco
607                                    Puno P. San Roman Near Tinco Palca
618                                              Puno P. Lampa Saraccocha
```

```
    lat lon coordUncertaintyM  alt institution
606  NA  NA              <NA> 4000      PER001
607  NA  NA              <NA> 4000      PER001
618  NA  NA              <NA> 4100      PER001
                 collection catalogNumber
606 CIP - Potato collection    CIP-762165
607 CIP - Potato collection    CIP-761962
618 CIP - Potato collection    CIP-762376
```

We recommend using a tool like BioGeomancer: `http://bg.berkeley.edu/latest` (Guralnick *et al.*, 2006) to georeference textual locality descriptions. An important feature of BioGeomancer is that it attempts to capture the uncertainty associated with each georeference (Wieczorek *et al.*, 2004). The dismo package has a function `biogeomancer` that you can use for this, and that we demonstrate below, but its use is generally not recommended because you really need a detailed map interface for accurate georeferencing.

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.

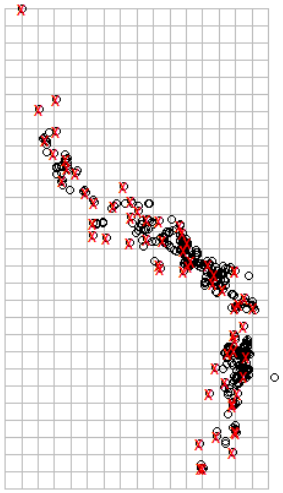
```
> georef$cloc[4]

[1] "Ayacucho P. Huamanga Minas Ckucho, Peru"

> b <- try(  geocode(georef$cloc[4]) )
> b

                           originalPlace
1 Ayacucho P. Huamanga Minas Ckucho, Peru
2 Ayacucho P. Huamanga Minas Ckucho, Peru
                           interpretedPlace longitude
1 Ayacucho, Buenos Aires Province, Argentina -58.48297
2                             Ayacucho, Peru -74.21582
   latitude      xmin      xmax      ymin      ymax
1 -37.15027 -58.49898 -58.46696 -37.15916 -37.14137
2 -13.15816 -74.24456 -74.18004 -13.19747 -13.11926
  uncertainty
1        1731
2        5372
```

Before using the geocode function it is best to write the records to a table and "clean" them in a spreadsheet. Cleaning involves traslation, expanding abbreviations, correcting misspellings, and making duplicates exactely 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 origninal data. For the geocoding it would be better to use the biogeomancer browser interface as it allows for much more (and necessary) interaction and editing.

2.5 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.

2.6 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 3

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).

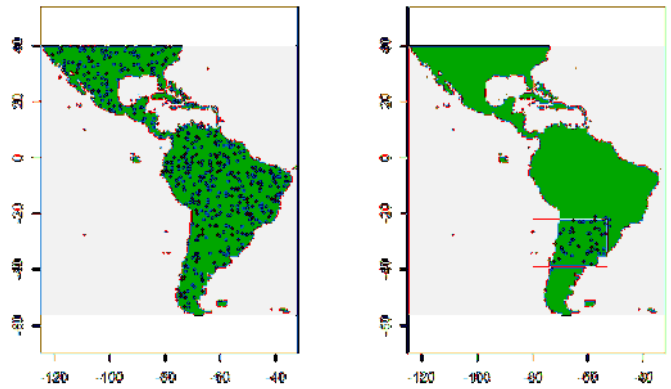
`dismo` has a function to sample random points (background data) from a study area. You can use a 'mask' to exclude area with no data `NA`, e.g. areas not on land. You can use an 'extent' to further restrict the area from which random locations are drawn. In the example below, we first get the list of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

```
> # 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
> # 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=lonlat')
```

We first create a 'cricles' 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)
```

Now we use the `rgeos` library to 'dissolve' the circles (remove boundaries were circles overlap).

```
> library(rgeos)
> pol <-  gUnaryUnion(x@polygons)
```

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)
> # get unique cells
> cells <- cellFromXY(mask, samp1)
> length(cells)
```

```
[1] 250
```

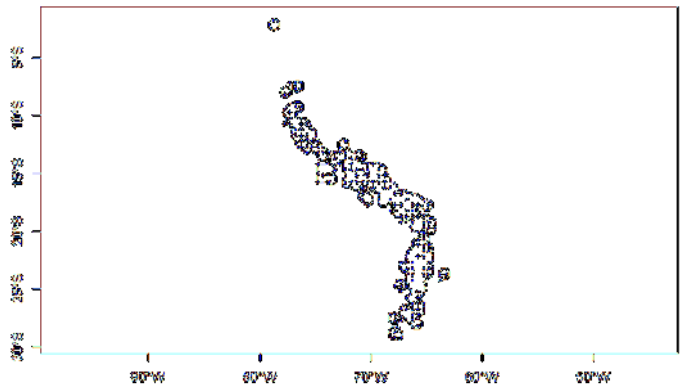
```
> cells <- unique(cells)
> length(cells)
```

```
[1] 157
```

```
> xy <- xyFromCell(mask, cells)
```

Plot to inspect the results:

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



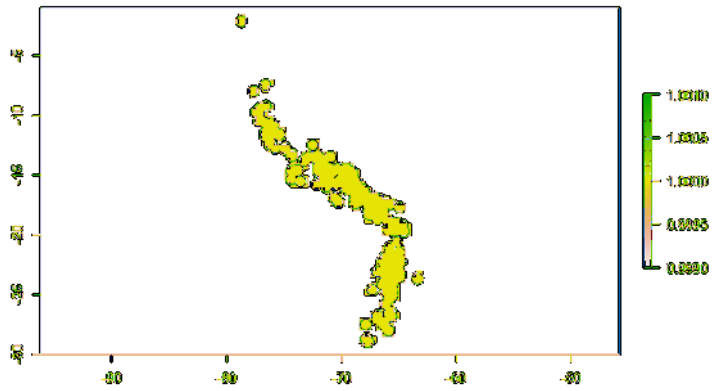
Note that the blue points are not all within the polygons (circles), as they now represent the centers of the selected cells from mask. We could choose to select only those cells that have their centers within the circles, using the overlay function.

```
> o <- overlay(x@polygons, SpatialPoints(xy))
> 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)
```



Chapter 4

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 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 `raster` package 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.

```
> files <- list.files(path=paste(system.file(package="dismo"),
+               '/ex', sep=''), pattern='grd', full.names=TRUE )
> # The above finds all the files with extension "grd" in the
> # examples ("ex") directory of the dismo package. You do not
> # need such a complex statement to get your own files.
> files

[1] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio1.grd"
[2] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio12.grd"
[3] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio16.grd"
[4] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio17.grd"
```

```
[5] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio5.grd"
[6] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio6.grd"
[7] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio7.grd"
[8] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/bio8.grd"
[9] "d:/temp/RtmpEdDJdw/Rinstda461ac34fd/dismo/ex/biome.grd"
```

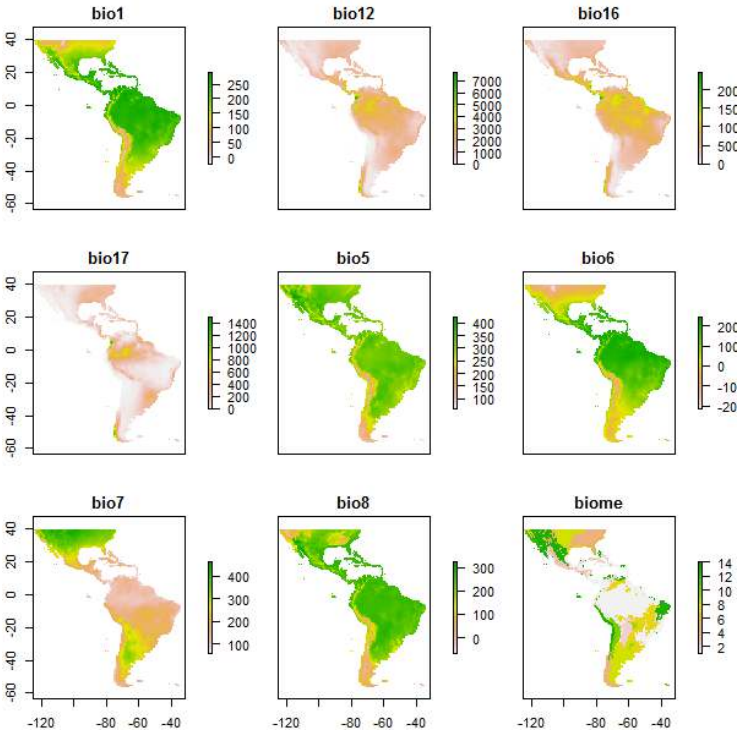
```
> predictors <- stack(files)
> predictors
```

```
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)
coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
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"
[8] "bio8"  "biome"
```

```
> plot(predictors)
```

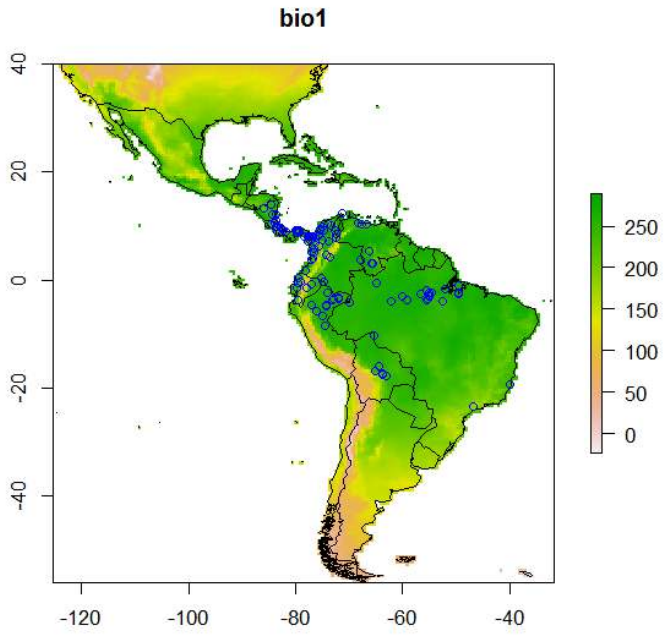


We can also make a plot of a single layer in a RasterStack, and plot some additional data on top of it. First get the world boundaries and the bradypus data:

```
> 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 (`http://www.worldclim.org`, Hijmans *et al.*, 2004) and 'terrestrial biome' data from the WWF. (`http://www.worldwildlife.org/`

science/data/item1875.html, 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  151  1434   434   277  316  -20  336   79     4
612  0  255  2229   970    91  331  178  153  254     1
613  0  249  1972   952    13  349  140  209  249     1
614  0  252  2793   975   528  308  198  110  246     1
615  0  119   280    89    57  280  -26  306   93     8
616  0  251  1696   672   158  320  194  126  246     2
```

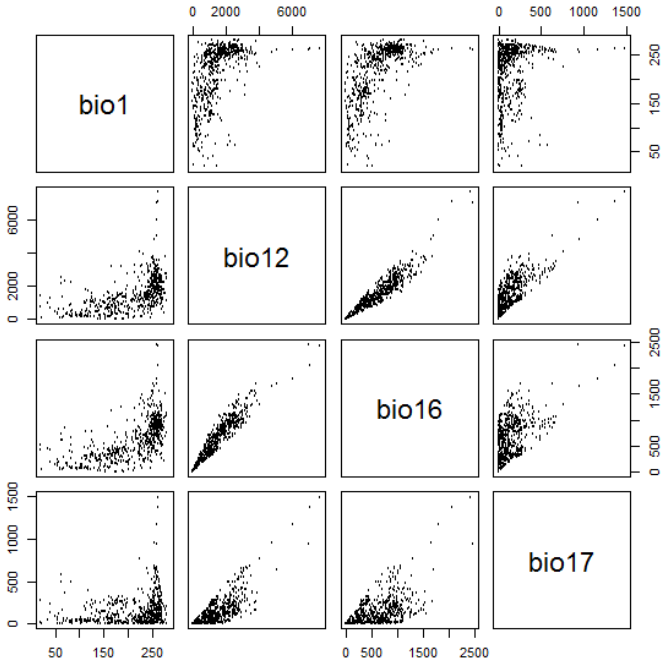
```
> summary(sdmdata)
```

```
       pb              bio1           bio12
 Min.   :0.0000   Min.   : 20.0   Min.   :   1.0
 1st Qu.:0.0000   1st Qu.:175.0   1st Qu.: 792.5
 Median :0.0000   Median :242.5   Median :1438.0
 Mean   :0.1883   Mean   :214.6   Mean   :1572.2
 3rd Qu.:0.0000   3rd Qu.:260.0   3rd Qu.:2233.0
 Max.   :1.0000   Max.   :282.0   Max.   :7682.0

     bio16           bio17            bio5
 Min.   :   1.0   Min.   :   0.0   Min.   : 89.0
 1st Qu.: 328.8   1st Qu.:  38.0   1st Qu.:303.8
 Median : 648.0   Median : 113.5   Median :319.5
 Mean   : 647.0   Mean   : 157.3   Mean   :309.9
 3rd Qu.: 916.0   3rd Qu.: 221.0   3rd Qu.:331.0
 Max.   :2458.0   Max.   :1496.0   Max.   :410.0

      bio6            bio7            bio8
 Min.   :-152.0   Min.   : 68.0   Min.   : -5.0
 1st Qu.:  44.0   1st Qu.:118.8   1st Qu.:217.8
 Median : 156.5   Median :157.5   Median :251.0
 Mean   : 120.8   Mean   :189.1   Mean   :224.7
 3rd Qu.: 201.0   3rd Qu.:246.2   3rd Qu.:261.2
 Max.   : 231.0   Max.   :461.0   Max.   :323.0

     biome
 1       :276
 13      : 79
 7       : 64
 8       : 61
 2       : 52
 (Other): 83
 NA's    :  1
```

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.* (2011) for a discussion of methods to remove colinearity.

```
> # pairs plot of the values of the climate data
> # at the bradypus occurrence sites.
> pairs(sdmdata[,2:5], cex=0.1, fig=TRUE)
```



Part II

Model fitting, prediction, and evaluation

Chapter 5

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' that as generated in the previous chapter.

```
> 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
-0.88043  -0.23410  -0.09852   0.07838   0.93738

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.1584718  0.1096895   1.445  0.14905
bio1         0.0012216  0.0003884   3.145  0.00174 **
```

```
bio5        -0.0014729  0.0004674  -3.152  0.00170 **
bio12        0.0001426  0.0000170   8.387 3.44e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1180497)

    Null deviance: 94.156  on 615  degrees of freedom
Residual deviance: 72.246  on 612  degrees of freedom
AIC: 437.94

Number of Fisher Scoring iterations: 2
```

```
> m2 = glm(pb ~ ., data=sdmdata)
> m2
```

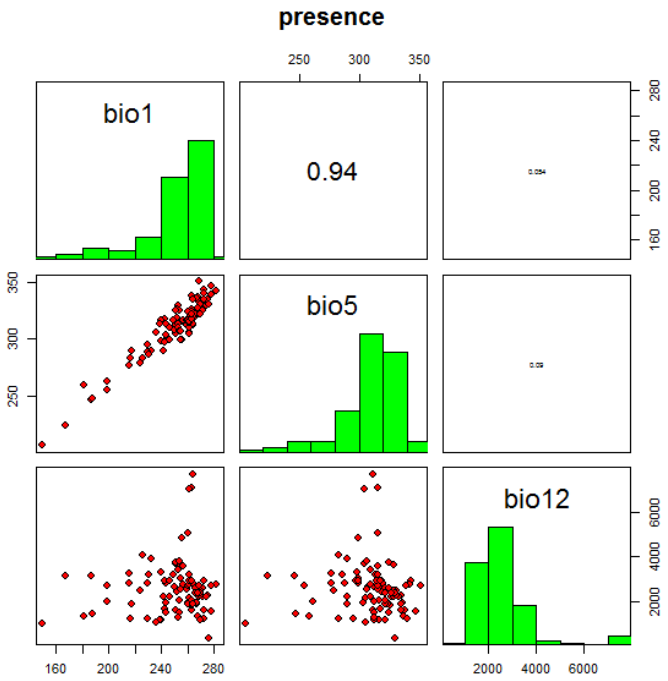
```
Call:  glm(formula = pb ~ ., data = sdmdata)

Coefficients:
(Intercept)         bio1        bio12        bio16
  0.3238958    0.0001823    0.0004699   -0.0006736
     bio17         bio5         bio6         bio7
 -0.0007612    0.0116904   -0.0126511   -0.0130362
       bio8       biome2       biome3       biome4
  0.0002934   -0.1008628   -0.0983808   -0.1623703
     biome5       biome7       biome8       biome9
 -0.0909717   -0.2241094   -0.0625490    0.0068173
    biome10      biome11      biome12      biome13
 -0.1207744   -0.3905900   -0.0698085    0.0152869
    biome14
 -0.2293511

Degrees of Freedom: 614 Total (i.e. Null);  594 Residual
  (1 observation deleted due to missingness)
Null Deviance:            94.12
Residual Deviance: 66.44         AIC: 420.7
```

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
  (... ...  ...)

> pairs(bc)
```



Chapter 6

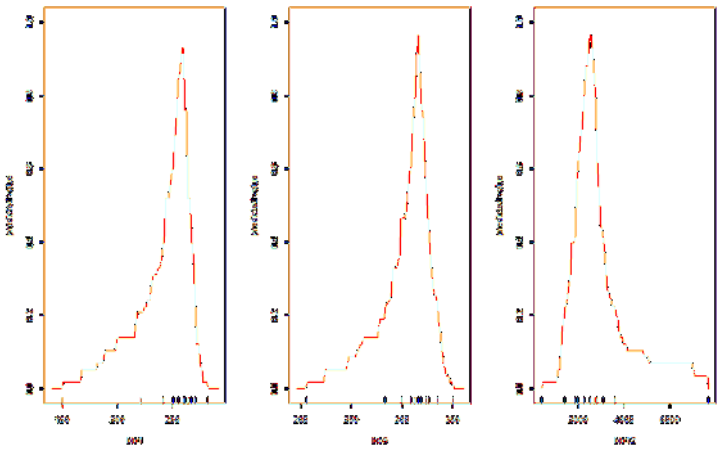
Model prediction

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)
> 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
0.2045259 0.4005016 0.2180797

> 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)
```

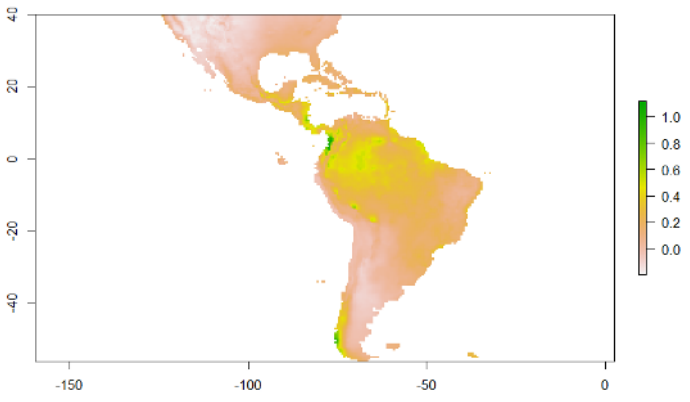


In most cases, howver, the purpose is SDM is to create a map of suitability scores. We can do that by providing the predict function with a Raster* object and a model object. As long as the variable names in the model object are available as layers (layerNames) in the Raster* object.

```
> names(predictors)
```

```
[1] "bio1"  "bio12" "bio16" "bio17" "bio5"  "bio6"  "bio7"
[8] "bio8"  "biome"
```

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



Chapter 7

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.

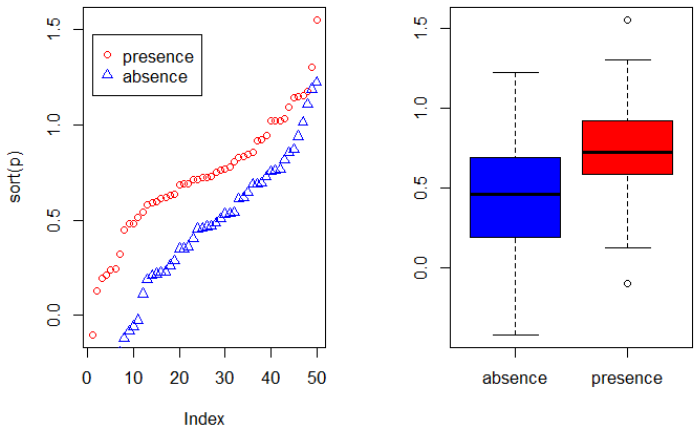
Different measures can be used to evaluate the quality of a prediction (Fielding and Bell, 1997, Liu et al., 2011; and Potts and Elith (2006) for abundance data), perhaps depending on the goal of the study. Many measures for evaluating models based on presence-absence or presence-only data are 'threshold 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'))
```

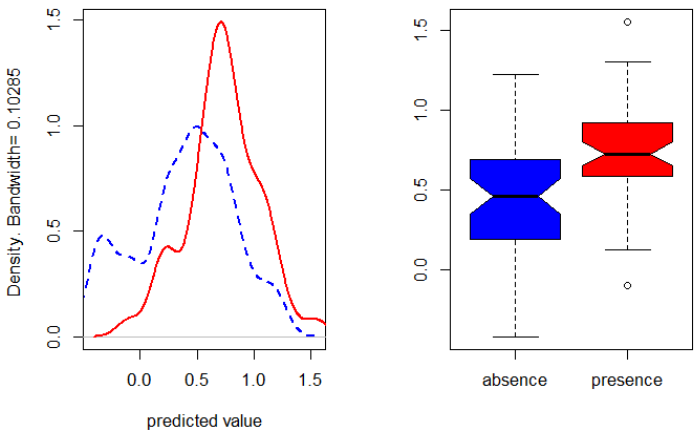


We created two variables with random normally distributed values, but with different mean and standard deviation. The two variables clearly have different distributions, and the values for 'presence' tend to be higher than for 'absence'. Here is how you can compute the correlation coefficient and the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

```
> group = c(rep(1, length(p)), rep(0, length(a)))
> cor.test(comb, group)$estimate

      cor
0.3996023

> mv <- wilcox.test(p,a)
> auc <- as.numeric(mv$statistic) / (length(p) * length(a))
> auc

[1] 0.7252
```

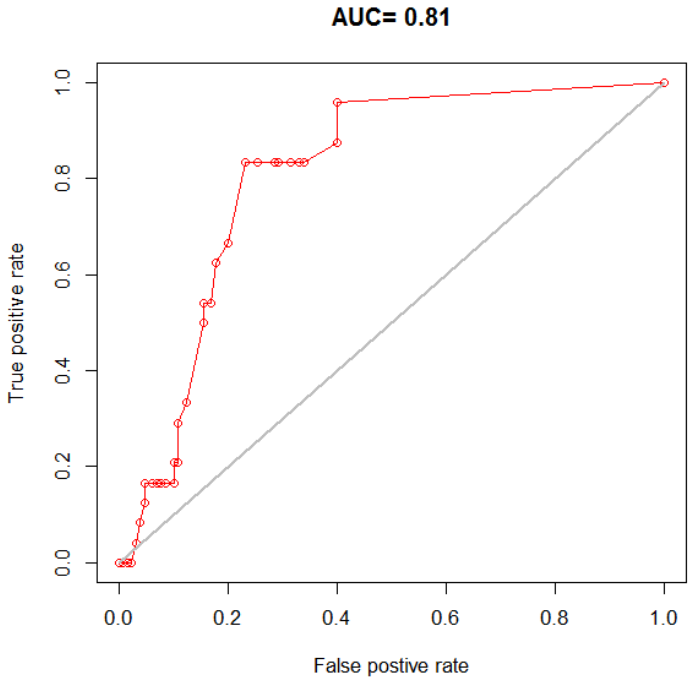
Below we show how you can compute these, and other statistics more conveniently, with the `evaluate` function in the dismo package. See ?evaluate for info on additional evaluation measures that are available. ROC/AUC can also be computed with the `ROCR` package.

```
> e <- evaluate(p=p, a=a)
> class(e)

[1] "ModelEvaluation"
attr(,"package")
[1] "dismo"

> e

class          : ModelEvaluation
n presences    : 50
n absences     : 50
AUC            : 0.7252
cor            : 0.3996023
max TPR+TNR at : 0.5427451

> par(mfrow=c(1, 2))
> density(e)
> boxplot(e, col=c('blue', 'red'))
```



Now 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    : 23
n absences     : 131
AUC            : 0.7293395
cor            : 0.2075071
max TPR+TNR at : 0.02140538

> 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] 4 5 3 3 3 1 3 5 5 1
> unique(group)
[1] 4 5 3 1 2
```

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)" (this is sometimes uses as a threshold for setting cells to presence or absence).

```
> auc <- sapply( e, function(x){slot(x, 'auc')} )
> auc

[1] 0.7786957 0.7900000 0.7302083 0.7526957 0.7428696

> mean(auc)

[1] 0.7588938

>  sapply( e, function(x){ x@t[which.max(x@TPR + x@TNR)] } )

[1] 0.03215806 0.12893226 0.04337826 0.03215806 0.03215806

> # equivalent
> # sapply( e, function(x){ x@t[which.max(x@TPR + x@TNR)] } )
```

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".

```
> nr <- nrow(bradypus)
> s <- sample(nr, 0.25 * nr)
> pres_train <- bradypus[-s, ]
> pres_test <- bradypus[s, ]
> nr <- nrow(backgr)
> 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
0.06838498
```

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] 1.003341
```

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            : 0.7816552
cor            : 0.2953961
max TPR+TNR at : 0.04587701

> evaluate(bc, p=pres_test_pwd, a=back_test_pwd, x=predictors)

class          : ModelEvaluation
n presences    : 15
n absences     : 15
AUC            : 0.3933333
cor            : -0.1373986
max TPR+TNR at : 0.4826586
```

Part III

Modeling methods

Chapter 8

Types of algorithms & 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/backround data. An entirely different class of models consists of models that only, or primarily, use the geographic location of known occurences, 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.

Let's first recreate the data we have used so far, such that you can step into the code starting here:

```
> files <- list.files(path=paste(system.file(package="dismo"),
+               '/ex', sep=''), pattern='grd', full.names=TRUE )
> predictors <- stack(files)
> file <- paste(system.file(package="dismo"), "/ex/bradypus.csv", sep="")
> 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.

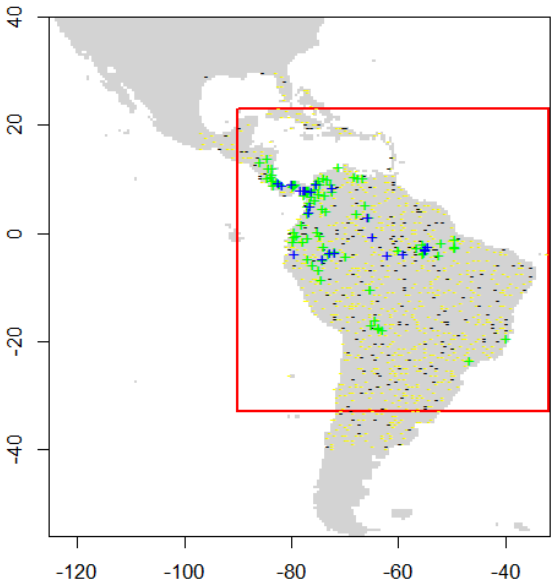
```
> group <- kfold(bradypus, 5)
> pres_train <- bradypus[group != 1, ]
> pres_test <- bradypus[group == 1, ]
```

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'.

```
> 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')
```


40
20
0
-20
-40
-120
-100
-80
-60
-40

Chapter 9

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.

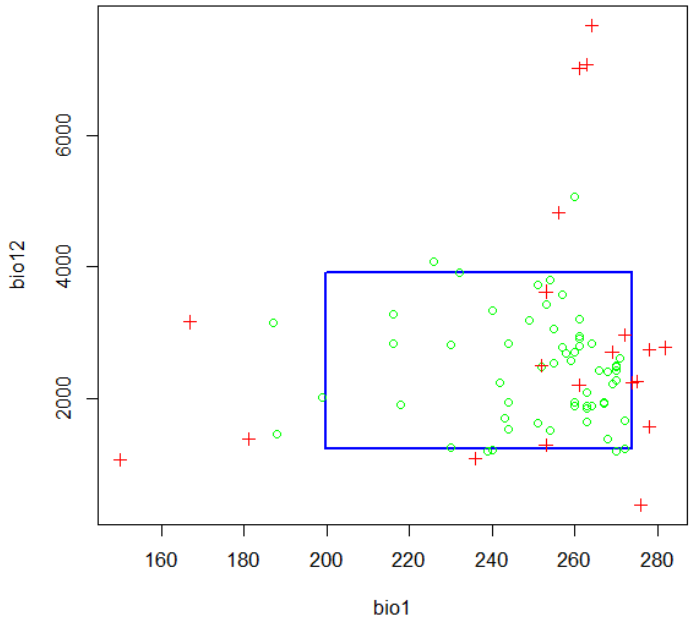
9.1 Bioclim

The BIOCLIM algorithm has been extensively used for species distribution modeling. BIOCLIM is a classic 'climate-envelope-model'. 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            : 0.7881522
cor            : 0.303771
max TPR+TNR at : 0.09667419
```

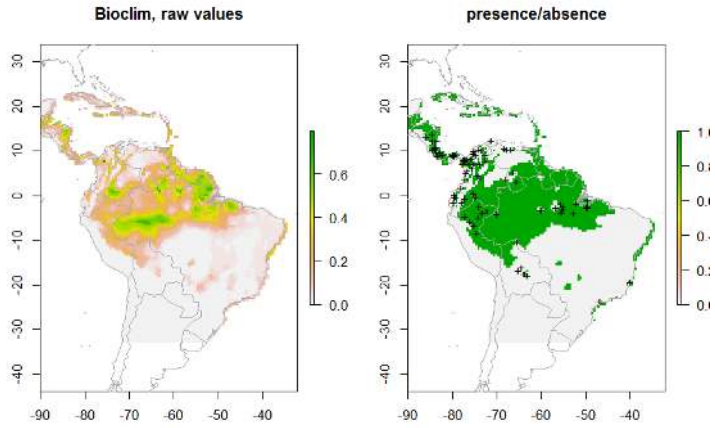
Find a threshold

```
> tr <- threshold(e, 'spec_sens')
> tr
```

```
[1] 0.09667419
```

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)
coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
data source : in memory
names       : layer
values      : 0, 0.7956989  (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.).

9.2 Domain

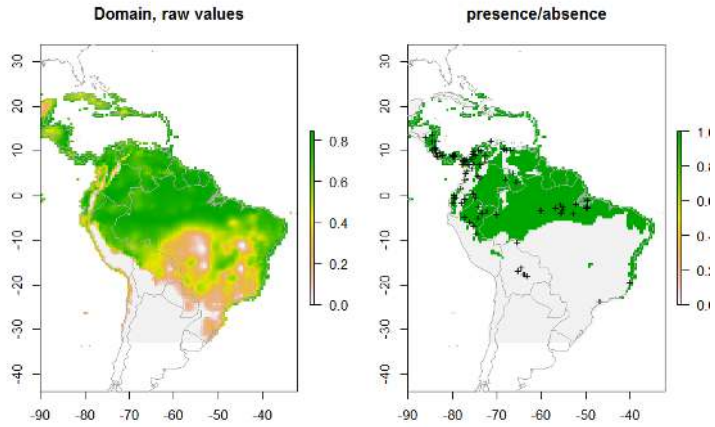
The Domain algorithm (Carpenter *et al.* 1993) has been extensively used for species distribution modeling. It did not perform very well in a model comparison (Elith *et al.* 2006) and very poorly when assessing climate change effects (Hijmans and Graham, 2006). The Domain algorithm computes the Gower 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.

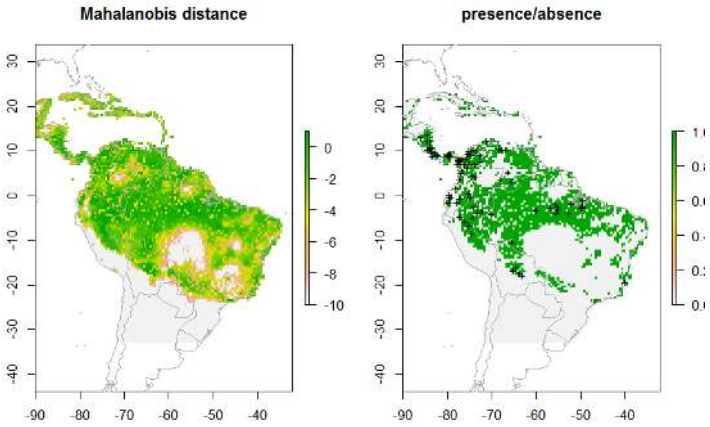
```
> dm <- domain(pred_nf, pres_train)
> e <- evaluate(pres_test, backg_test, dm, pred_nf)
> e

class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.7566304
cor            : 0.2660474
max TPR+TNR at : 0.6940046

> 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='+')
```



9.3 Mahalanobis

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            : 0.8521739
cor            : 0.1401341
max TPR+TNR at : -1.790251

> 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='+')
```


Mahalanobis distance
presence/absence
30
20
10
0
-10
-20
-30
-40
-90
-80
-70
-60
-50
-40
0
-2
-4
-6
-8
-10

Chapter 10

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  243  1693   775   186  318  150  168  264     1
6  1  252  2501  1081   280  326  154  172  270     1

> 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)
```

10.1 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
-1.62737  -0.48920  -0.25822  -0.06483   2.66953

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.1803848  1.5378611   1.418  0.15625
bio1         0.0286160  0.0532146   0.538  0.59075
bio5         0.0616622  0.2584455   0.239  0.81142
bio6        -0.1152667  0.2595235  -0.444  0.65694
bio7        -0.1264023  0.2586404  -0.489  0.62504
bio8         0.0215463  0.0241154   0.893  0.37161
bio12        0.0019683  0.0007005   2.810  0.00495 **
bio16       -0.0016414  0.0014694  -1.117  0.26397
bio17       -0.0052749  0.0016206  -3.255  0.00113 **
---
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: 454.33  on 884  degrees of freedom
AIC: 472.33

Number of Fisher Scoring iterations: 7

> coef(gm1)
```

```
 (Intercept)         bio1         bio5         bio6
 2.180384805  0.028616011  0.061662150 -0.115266711
        bio7         bio8        bio12        bio16
-0.126402323  0.021546255  0.001968337 -0.001641381
       bio17
-0.005274945
```

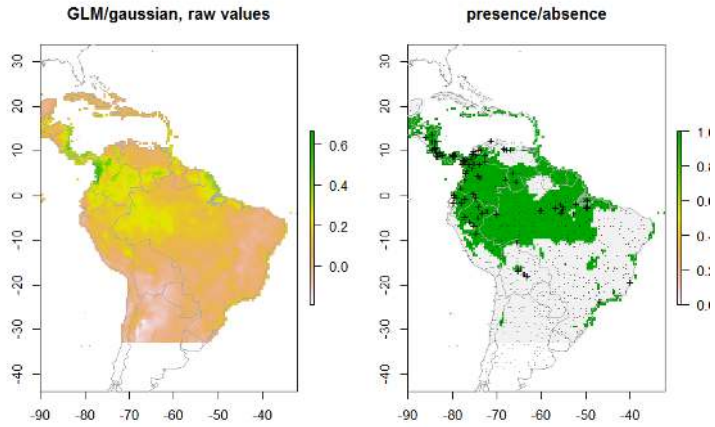
```
> gm2 <- glm(pa ~ bio1+bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17,
+             family = gaussian(link = "identity"), data=envtrain)
> evaluate(testpres, testbackg, gm1)
```

```
class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.8530435
cor            : 0.3270563
max TPR+TNR at : -2.244521
```

```
> ge2 <- evaluate(testpres, testbackg, gm2)
> ge2
```

```
class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.8378261
cor            : 0.3870294
max TPR+TNR at : 0.1117668
```

```
> 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)
```



10.2 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.

Chapter 11

Machine learning methods

There is a variety of machine learning (sometimes referred to data mining) methods in R . For a long time there have been packages to do Artifical Neural Networks (ANN) and Classification and Regression Trees (CART). More recent methods include 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.

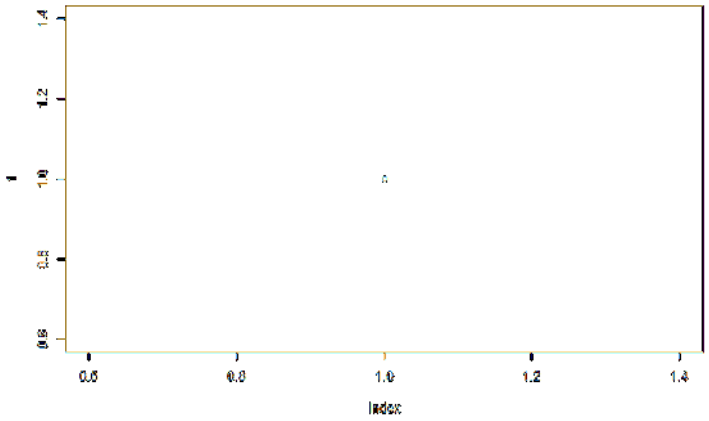
11.1 Maxent

MaxEnt (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 standalone Java program. Dismo has a function 'maxent' that communicates with this program. To use it you must first download the program from `http://www.cs.princeton.edu/~schapire/maxent/`. Put the file 'maxent.jar' in the 'java' folder of the 'dismo' package. That is the folder returned by `system.file("java", package="dismo")`. Please note that this program (`maxent.jar`) cannot be redistributed or used for commercial purposes.

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.

```
> # checking if the jar file is present. If not, skip this bit
> jar <- paste(system.file(package="dismo"), "/java/maxent.jar", sep='')
> if (file.exists(jar)) {
+         xm <- maxent(predictors, pres_train, factors='biome')
+         plot(xm)
+ } else {
+     cat('cannot run this example because maxent is not available')
+         plot(1)
+ }
```

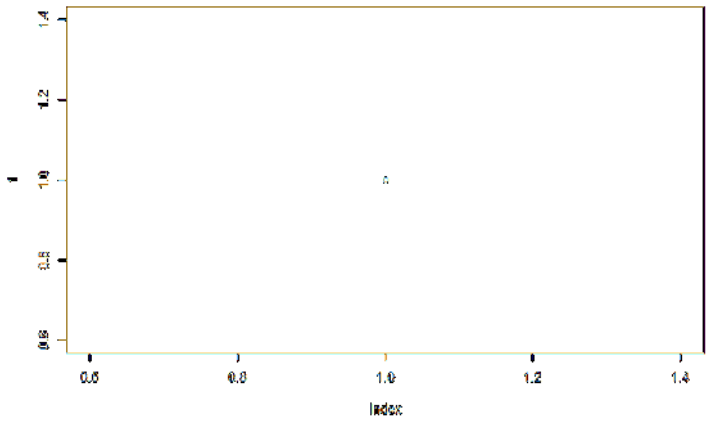
```
cannot run this example because maxent is not available
```



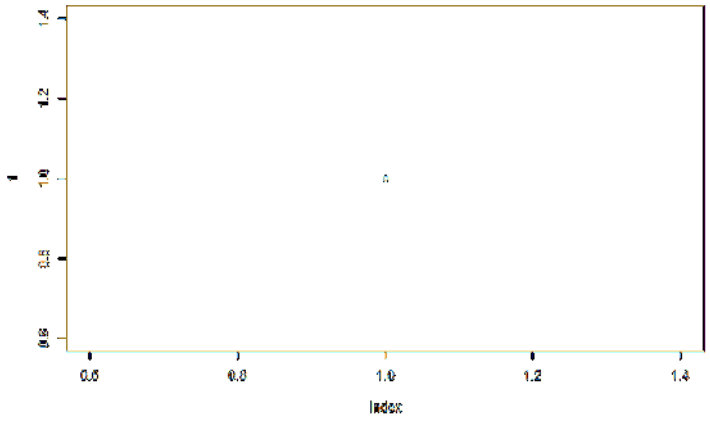
A response plot:

```
> if (file.exists(jar)) {
+         response(xm)
+ } else {
+     cat('cannot run this example because maxent is not available')
+         plot(1)
+ }
```

```
cannot run this example because maxent is not available
```




```
> if (file.exists(jar)) {
+         e <- evaluate(pres_test, backg_test, xm, predictors)
+         e
+         px <- predict(predictors, xm, ext=ext, progress='')
+         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='+')
+ } else {
+         plot(1)
+ }
```



11.2 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 .

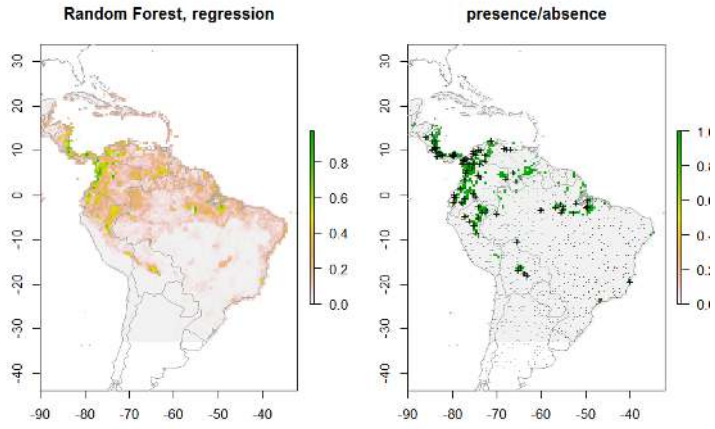
The article by Elith, Leathwick and Hastie (2009) describes the use of BRT in the context of species distribution modeling. Their article is accompanied by a number of R functions and a tutorial that have been slightly adjusted and incorporated into the 'dismo' package. These functions extend the functions 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 a vignette to the dismo package. You can access it via the index of the help pages, or with this command: `vignette('gbm', 'dismo')`

11.3 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)
> model <- pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17
> rf1 <- randomForest(model, data=envtrain)
> 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            : 0.8621739
cor            : 0.521359
max TPR+TNR at : 0.2918667
```

```
> 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)
```



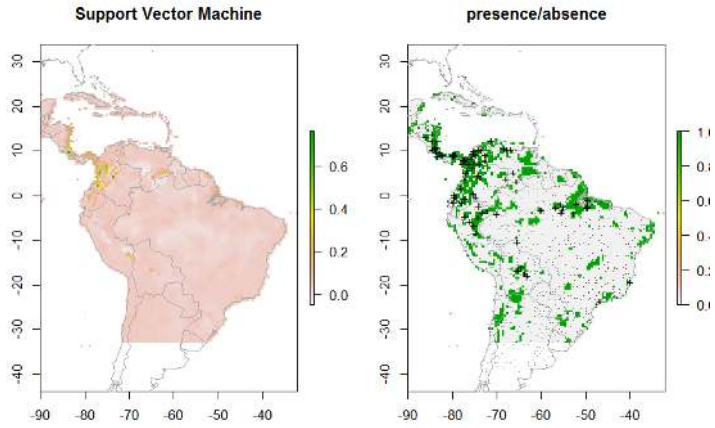
11.4 Support Vector Machines

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)
> svm <- ksvm(pa ~ bio1+bio5+bio6+bio7+bio8+bio12+bio16+bio17, data=envtrain)
```

```
Using automatic sigma estimation (sigest) for RBF or laplace kernel

> esv <- evaluate(testpres, testbackg, svm)
> esv

class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.7276087
cor            : 0.4300625
max TPR+TNR at : 0.03073177

> 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)
```



Chapter 12

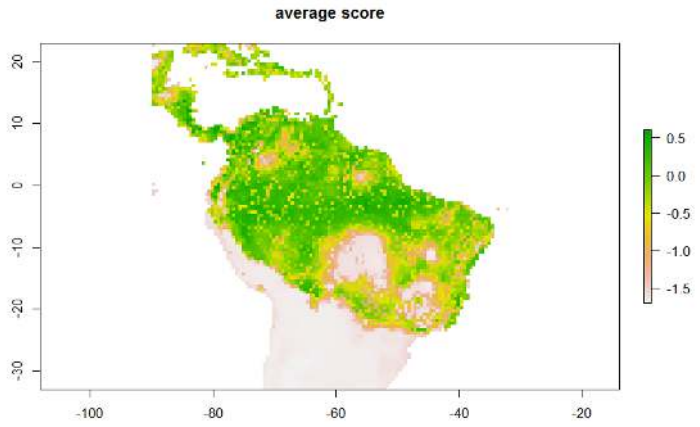
Combining model predictions

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)
```

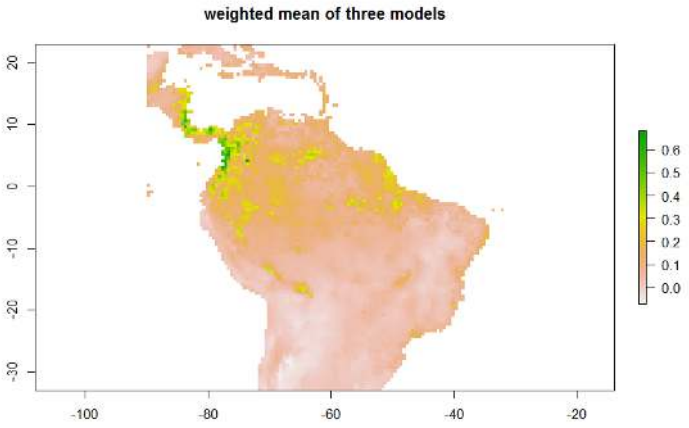
Now we can compute the simple average:

```
> 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')
```



Chapter 13

Geographic models

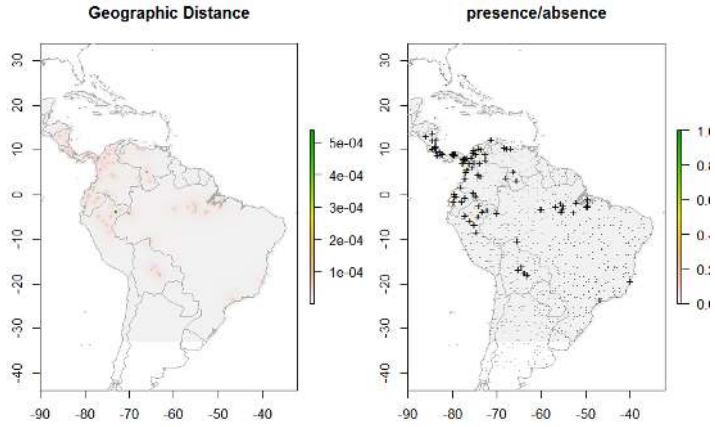
The 'geographic models' described here are not commonly used in species distribution modeling. They use the geographic location of known occurences, and do not rely on the values of predictor variables at these locations. We are exploring their use in comparing and contrasting them with the other approaches (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 (Colwel and Rangel, 2009). Below we show examples of these different types of models.

13.1 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.

```
> # 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            : 0.933913
cor            : 0.5831666
max TPR+TNR at : 9.199e-05

> par(mfrow=c(1,2))
> plot(ds, main='Geographic Distance')
```

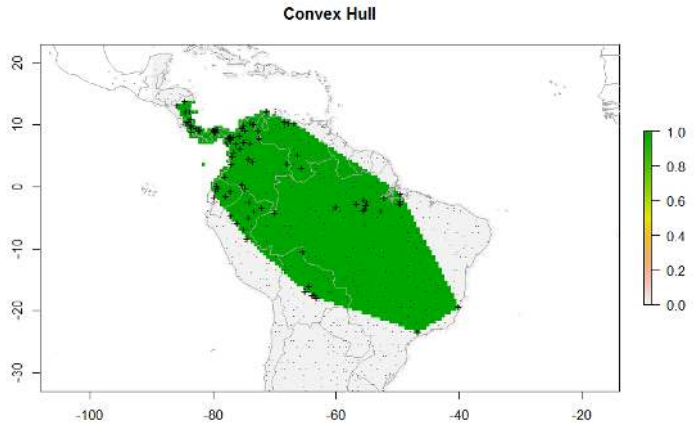
```
> plot(wrld_simpl, add=TRUE, border='dark grey')
> tr <- threshold(e, 'spec_sens')
> plot(ds > tr, main='presence/absence')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> points(pres_train, pch='+')
> points(backg_train, pch='-', cex=0.25)
```



13.2 Convex hulls

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

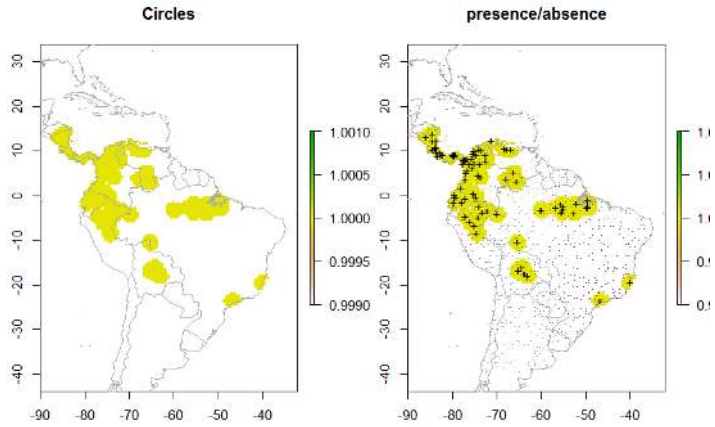
```
> hull <- convHull(pres_train, lonlat=TRUE)
> e <- evaluate(hull, p=pres_test, a=backg_test)
> e

class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.7282609
cor            : 0.2789329
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)
```



13.3 Circles

This model draws circles around all 'presence' points.

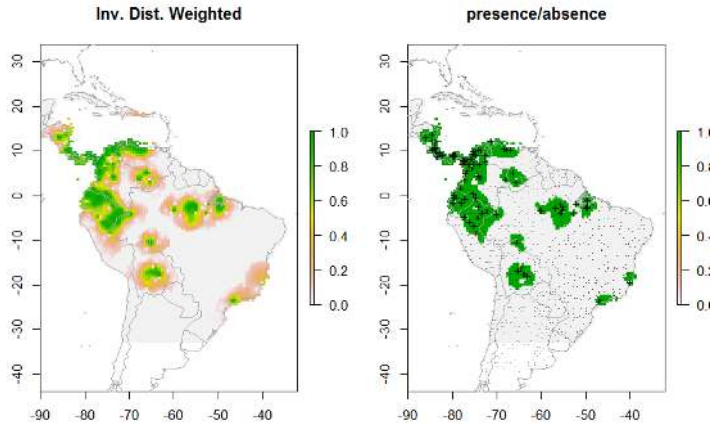
```
> circ <- circles(pres_train, lonlat=TRUE)
> pc <- predict(seamask, circ, mask=TRUE)
> e <- evaluate(circ, p=pres_test, a=backg_test)
> e

class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.8147826
cor            : 0.41591
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)
```



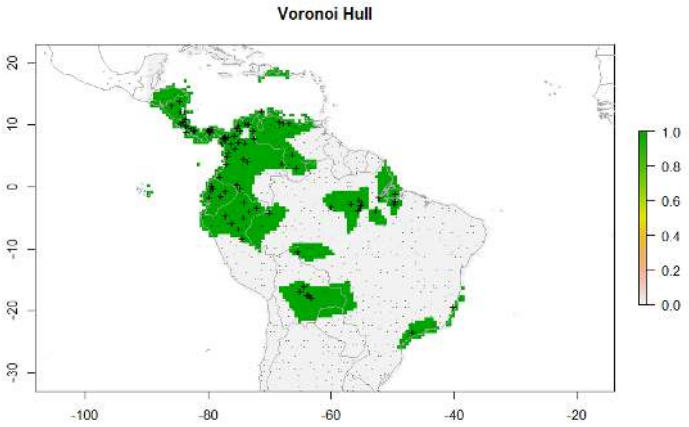
13.4 Presence/absence

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 pacakge).

```
> idwm <- geoIDW(p=pres_train, a=data.frame(back_train))
> e <- evaluate(idwm, p=pres_test, a=backg_test)
> e

class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.9102174
cor            : 0.5398107
max TPR+TNR at : 0.2740503

> 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')
> 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)

     PLEASE NOTE:  The components "delsgs" and "summary" of the
     object returned by deldir() are now DATA FRAMES rather than
     matrices (as they were prior to release 0.0-18).
     See help("deldir").

     PLEASE NOTE: The process that deldir() uses for determining
     duplicated points has changed from that used in version
     0.0-9 of this package (and previously). See help("deldir").

> e <- evaluate(vorm, p=pres_test, a=backg_test)
> e

class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.4925
cor            : -0.03960028
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='+')
> points(backg_train, pch='-', cex=0.25)
```


Voronoi Hull
20
10
0
-10
-20
-30
-100
-80
-60
-40
-20
1.0
0.8
0.6
0.4
0.2
0.0

Part IV

Additional topics

Chapter 14

Model transfer in space and time

14.1 Transfer in space

14.2 Transfer in time: climate change

Chapter 15

To do list

You can ignore this chapter, it is the authors' to-do list.

There are many sophistications that are required by the realities that (a) there are multiple end uses of models, and (b) there are numerous issues with ecological data that mean that the assumptions of the standard methods don't hold. Could include:

- spatial autocorrelation
- imperfect detection
- mixed models (for nested data, hierarchical stuff)
- Bayesian methods
- resource selection functions
- measures of niche overlap, linked to thoughts about niche conservatism
- link to phylogeography
- additional predictors including remote sensing variables, thinking about extremes
- species that don't "mix" with grids – freshwater systems etc.
- quantile regression
- model selection literature (AIC etc etc)
- multispecies modeling: Mars, gdm
- SDMTools
- Dealing with uncertainty using uncertainty field in georeferences. How to target the "important" uncertainties (will vary with the application), an example of partial plots with standard errors, and predicting the upper and lower bounds; the idea of testing sensitivity to decisions made in the modeling process (including dropping out points etc etc).

Part V

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.

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. (2011 in review) Collinearity: a review of methods to deal with it and a simulation study evaluating their performance.

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. http://dx.doi.org/10.1146/annurev.ecolsys.110308.120159

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. http://dx.doi.org/10.1111/j.2006.0906-7590.04596.x

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. http://dx.doi.org/10.1111/j.1472-4642.2010.00725.x

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. http://www-stat.stanford.edu/~jhf/ftp/trebst.pdf)

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. http://dx.doi.org/10.1371/journal.pbio.0040381

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) http://www-stat.stanford.edu/~tibs/ElemStatLearn/

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. http://dx.doi.org/10.1111/j.1365-2486.2006.01256.x

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. http://dx.doi.org/10.1002/joc.1276

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). http://www.jstatsoft.org/v15/i09/

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.