Species distribution modeling with R

Robert J. Hijmans and Jane Elith

May 6, 2011

Chapter 1

Introduction

This document is an introduction to species distribution modeling with R . Species distribution modeling (SDM) is also known under other names including envelope-modeling and (environmental or ecological) niche-modeling. In SDM, the following steps are usually taken: (1) locations of occurrence (and perhaps non-occurrence) of a species (or other phenomenon) are compiled. (2) values of environmental predictor variables (such as climate) at these locations are determined. (3) the environmental values are used to fit a model predicting likelihood of presence, or another measure such as abundance for the species. (4) The model is used to predict the likelihood of presence at all locations of an area of interest (and perhaps in a future climate).

We do not provide a general introduction to species distribution modeling itself. We assume that you are familiar with most of the concepts in this field. If in doubt, you could consult Richard Pearson's introduction to the subject: `http://biodiversityinformatics.amnh.org/index.php?section_id=111`, or the book by Janet Franklin (2009). You can also consult a recent review of the field by Elith and Leathwick (2009).

We also assume that you are already 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 the '`raster`' package. If you are not experienced with these, we recommend you read some tutorials or introductions. See, for instance, the Documentation section on the CRAN webpage (http://cran.r-project.org/) and the vignette for the '`raster`' package. When we present code we will give some hints on how to understand the code, if we think it might be 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 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 such as '`BIOMOD`)'. If you want to test, or build on, some of the examples presented here, make sure you have the latest versions of these libraries, and their dependencies, installed. If you are using a recent version of

R , you can do that with:

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

Part I

Data preparation

Data preparation 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) of the species of you interest. You also need to have accurate and relevant spatial 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 data are sufficiently accurate to adequately represent the species' distribution. 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. Further information on this general topic can be found in (citations, ..).

Chapter 2

Species occurrence data

Importing occurrence data into R is easy. But collecting, georeferencing, and cross-checking coordinate data is tedious. While we'll show you some useful data preparation steps you can do in R , it is necessary to use additional tools as well. 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. All methods do better if your occurrence data is unbiased and free of error (Graham et al., 2007) and you have have a relatively large number of records (Wisz et al., 2008).

2.1 Importing occurrence data

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

to do: introduce conventions: R code italics and preceded by > comments preceded by a hash (#)

if you haven't used the paste command before, it's worth familiarizing yourself with it (type ?paste in the command window). It's very useful. system.file inserts the file path to where it has installed dismo.

```
> library(dismo)

raster version 1.8-12 (11-April-2011)

> #this loads the dismo library
> filename <- paste(system.file(package="dismo"), '/ex/bradypus.csv', sep='')
> filename
```

```
[1] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bradypus.csv"
```

```
> bradypus <- read.table(filename, header=TRUE, sep=',')
> 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
```

```
> 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 or from a database (see e.g. the RODBC package). No matter how you do it, the objective is to get a matrix (or a data.frame) with at least 2 columns to hold the coordinates (typically longitude and latitude). A typical convention is to organize the coordinates columns so that longitude (x) is first and latitude, second. 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 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 else you can specify to this command, check out the help file (remember you can't access help files if the library is not loaded)

Many records may not have coordinates. Out of the 699 records that gbif returned (March 2010), there were only 54 records with coordinates.

```
> data(acaule)
> dim(acaule)
```

```
[1] 699  23
```

```
> acgeo <- subset(acaule, !is.na(lat) & !is.na(lon))
> dim(acgeo)
```

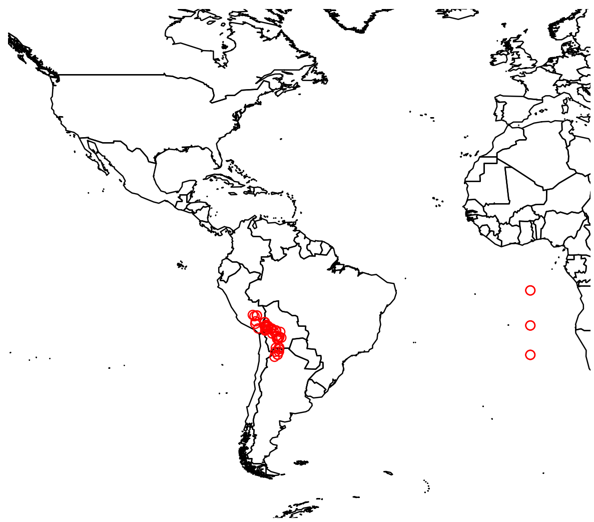
```
[1] 54 23
```

```
> acgeo[1:4, c(1:5,7:10)]
```

```
                  species continent   country   adm1    adm2      lat    lon
13         Solanum acaule      <NA>       BOL   <NA>    <NA> -18.8167 -65.90
426 Solanum acaule Bitter   America Argentina  Jujuy         -22.9000 -66.24
428 Solanum acaule Bitter   America   Bolivia La Paz Pacajes -17.4200 -68.85
429 Solanum acaule Bitter   America   Bolivia La Paz Pacajes -17.1200 -68.77
    coordUncertaintyM  alt
13                 NA 3960
426                NA 4050
428                NA 3811
429                NA 3800
```

Here is a simple way to make a map of the occurrence localities of *Solanum acaule*:

```
> library(maptools)
> data(wrld_simpl)
> plot(wrld_simpl, xlim=c(-130,10), ylim=c(-60,60))
> points(acgeo$lon, acgeo$lat, col='red')
```



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 read your own shapefile into R using the `readOGR` function in the `rgdal` package or the `readShapePoly` function in the `maptools` 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 Peru and Bolivia. Do you see any errors on the map? There are 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 (because they appear to be exactly South of London). In many data-bases you will find values that are 'zero' where 'no data' was intended.

The gbif function (with default arguments) removes records that have (0, 0) as coordinates, but not if one of the coordinates is zero. Let's see if we find them by searching for records with longitudes of zero (later we show you how to click on and identify outliers (to do))

Let's have a look at these records:

```
> lonzero = subset(acgeo, lon==0)
> lonzero[, 1:13]
```

```
                                species continent country adm1 adm2
544 Solanum acaule Bitter subsp. acaule      <NA>     BOL <NA> <NA>
551 Solanum acaule Bitter subsp. acaule      <NA>     BOL <NA> <NA>
567 Solanum acaule Bitter subsp. acaule      <NA>     PER <NA> <NA>
638 Solanum acaule Bitter subsp. acaule      <NA>     PER <NA> <NA>
640 Solanum acaule Bitter subsp. acaule      <NA>     ARG <NA> <NA>
641 Solanum acaule Bitter subsp. acaule      <NA>     ARG <NA> <NA>
                                            locality       lat lon
544                                            Llave -16.083333   0
551                                            Llave -16.083333   0
567                     km 205 between Puno and Cuzco  -6.983333   0
638                     km 205 between Puno and Cuzco  -6.983333   0
640 between Quelbrada del Chorro and Laguna Colorada -23.716667   0
641 between Quelbrada del Chorro and Laguna Colorada -23.716667   0
    coordUncertaintyM  alt institution collection catalogNumber
544                NA 3900         IPK  WKS 30050        304711
551                NA 3900         IPK         GB     WKS 30050
567                NA 4250         IPK  WKS 30048        304709
638                NA 4250         IPK         GB     WKS 30048
640                NA 3400         IPK  WKS 30027        304688
641                NA 3400         IPK         GB     WKS 30027
```

The records are from Bolivia (BOL), Peru (PER) and Argentina (ARG), confirming that coordinates are in error (it could have been that the coordinates were correct for a location in the Ocean, perhaps referring to a location a fish was caught rather than a place where S. acaule was collected).

2.2.1 duplicate records

Interestingly, another data quality issue is revealed above: each record occurs twice. This could happen because plant samples are often split and send to multiple herbariums. But in this case it seems that a single GBIF data provider (IPK) has these record duplicated in its database.

To do: provide code for checking for duplicates. Two issues: exact duplicates (lat / long identical) and duplicates on a per grid cell basis.

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)
> coordinates(acgeo) = ~lon+lat
> ov = overlay(acgeo, wrld_simpl)
> cntr = as.character(wrld_simpl@data$NAME[ov])
> which(is.na(cntr))

[1] 43 44 45 46 47 48

> i = which(cntr != acgeo@data$country)
> cbind(cntr, acgeo@data$country)[i,]

     cntr
[1,] "Bolivia" "BOL"
[2,] "Bolivia" "BOL"
[3,] "Peru"    "PER"
[4,] "Peru"    "PER"
[5,] "Peru"    "PER"
```

Note that the polygons that we used in the example above are not very precise, and they 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='PER', level=0)` to get the national borders of Peru. The overlay function returned indices (row numbers) that we stored in variable '`i`'. We used these in the next line to get the country for each point. Then we ask which countries are '`NA`' (i.e., points in oceans), and which countries have non matching names (in this case these are all caused by using abbreviations in stead of full names).

```
> acgeo  = acgeo[coordinates(acgeo)[,'lon'] < 0, ]
```

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] 89 23
```

```
> georef[1:3,1:13]

                                                     species continent country
30 Solanum acaule Bitter subsp. acaule (Juz.) Hawkes & Hjert.      <NA>     PER
42 Solanum acaule Bitter subsp. acaule (Juz.) Hawkes & Hjert.      <NA>     BOL
81 Solanum acaule Bitter subsp. acaule (Juz.) Hawkes & Hjert.      <NA>     ARG
   adm1 adm2                      locality lat lon coordUncertaintyM  alt
30 <NA> <NA> km 205 between Puno and Cuzco  NA  NA                NA 4250
42 <NA> <NA>                         Llave  NA  NA                NA 3900
81 <NA> <NA>                       da Pena  NA  NA                NA   NA
   institution collection catalogNumber
30      DEU159        DEU     WKS 30048
42      DEU159        DEU     WKS 30050
81      DEU159        DEU     WKS 30417
```

Among the first records is an old acquaintance. The record, with catalog number WKS 30048 was also in the set of records that had a longitude of zero degrees. This time it seems that it is served to gbif via another institution 'DEU' (I suspect that these duplicates occur because GBIF has records from aggregators such as EURISCO and national nodes, as well as from individual institutes).

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. We put the biogeomancer function into a 'try' function, to assure elegant error handling if the computer is not connected to the Internet.

```
> args(biogeomancer)

function (country = "", adm1 = "", adm2 = "", locality = "",
    singleRecord = TRUE, progress = "text")
NULL

> b = try(  biogeomancer('Peru', locality=lonzero$locality[3], progress='')  )
> b

   id lon lat coordUncertaintyM
NA  1  NA  NA                NA

> lonzero$lat[3]

[1] -6.983333
```

Note that the uncertainty (expressed in meters) is quite high, and that the latitude is rather different from the original latitude (whereas the original latitude might in fact be correct). With this much uncertainty, it is likely that you would choose not to use this point unless further information that allowed more refinement could be found.

2.5 Sampling bias

(Phillips et al., 2009)

Chapter 3

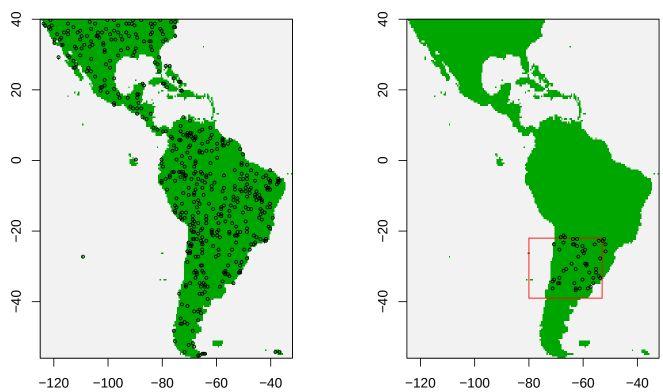
Absence and background points

Many of the early species distribution models, such as Bioclim and Domain are known as 'profile' methods because they only use 'presence' data. Hence they are known as presence-only methods. 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; and the 'maxent' algorithm is also closely related to logistic regression). 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, also referred to as 'random absence'or 'pseudo-absence' (though sometimes with varying meanings) may not always be *that* different from 'true absence' data. If you have a species with a range that is relatively small compared to the study area, there will only a few background points where the species is actually present. Moreover, the species might be absent (not observable) at these sites at a given time of sampling (depending on scale, detectability, ...). Despite the potential similarities between background and absence data, some researchers reserve a special definition for background data. For these (e.g. Phillips et al. 2009), background data 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. 'True' 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 (refercences).

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.

```
> files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''),
+                     pattern='grd', full.names=TRUE )
> mask <- raster(files[[1]])
> bg <- randomPoints(mask, 500 )
> par(mfrow=c(1,2))
> plot(!is.na(mask), legend=FALSE)
> points(bg, cex=0.5)
> # now with an 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)
```



**here could add info on random sampling "biased" by cell area.

**I think here or later we need a bit of commentary on how to choose the background extent for modeling. (i.e. pertinent to what is being answered, ecologically relevant to the species and the presence sample)..

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 and 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, and ASCII). Avoid ASCII files if you can, as they tend to considerably slow down processing speed. For any particular study the layers all should have the same spatial extent, resolution, and origin (if necessary, see the '`raster`' package to prepare your predictor variable data). The set of predictor variables (raster) can be used to make a '`RasterStack`', which can be thought of as 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] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio1.grd"
[2] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio12.grd"
[3] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio16.grd"
[4] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio17.grd"
[5] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio5.grd"
[6] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio6.grd"
```

```
[7] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio7.grd"
[8] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/bio8.grd"
[9] "/tmp/RtmpmLn4d8/Rinst1ef95355/dismo/ex/biome.grd"
```

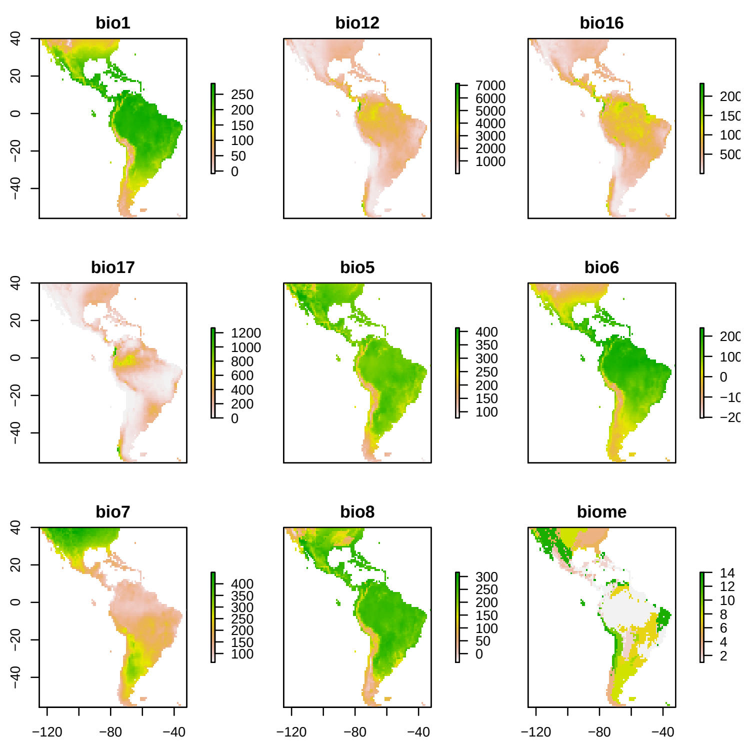
```
> predictors <- stack(files)
> predictors
```

```
class       : RasterStack
dimensions  : 192, 186, 9  (nrow, ncol, nlayers)
resolution  : 0.5, 0.5  (x, y)
extent      : -125, -32, -56, 40  (xmin, xmax, ymin, ymax)
projection  : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
min values  :  -23    0    0    0   61 -212   60  -66    1
max values  :  289 7682 2458 1496  422  242  461  323   14
```

```
> layerNames(predictors)
```

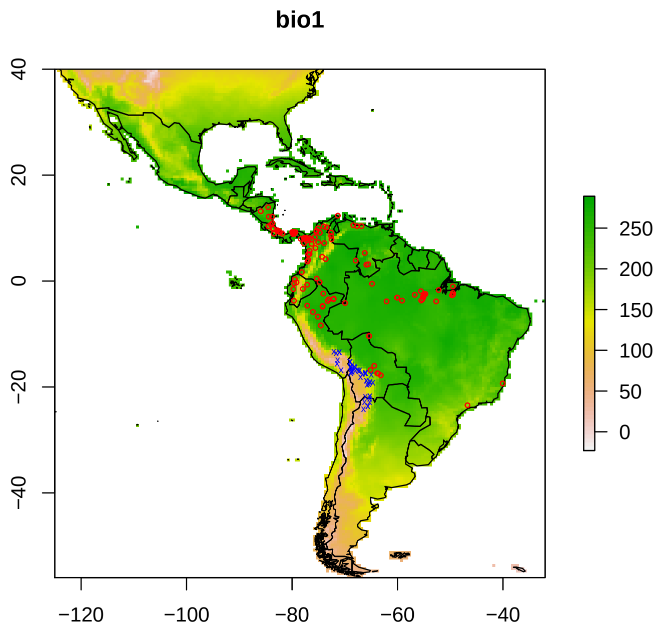
```
[1] "bio1"  "bio12" "bio16" "bio17" "bio5"  "bio6"  "bio7"  "bio8"  "biome"
```

```
> plot(predictors)
```



We can also make a plot of a single layer in a RasterStack, and plot some additional data on top of it:

```
> plot(predictors, 1)
> plot(wrld_simpl, add=TRUE)
> points(bradypus, col='red', cex=0.5)
> points(acgeo, col='blue', pch='x', cex=0.5)
```



The example above uses data representing 'bioclimatic variables' from the WorldClim database (`http://www.worldclim.org`, Hijmans et al., 2004) and 'terrestiral 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 (as well as other geographic data).

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 very straightfoward 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)
> 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)
```

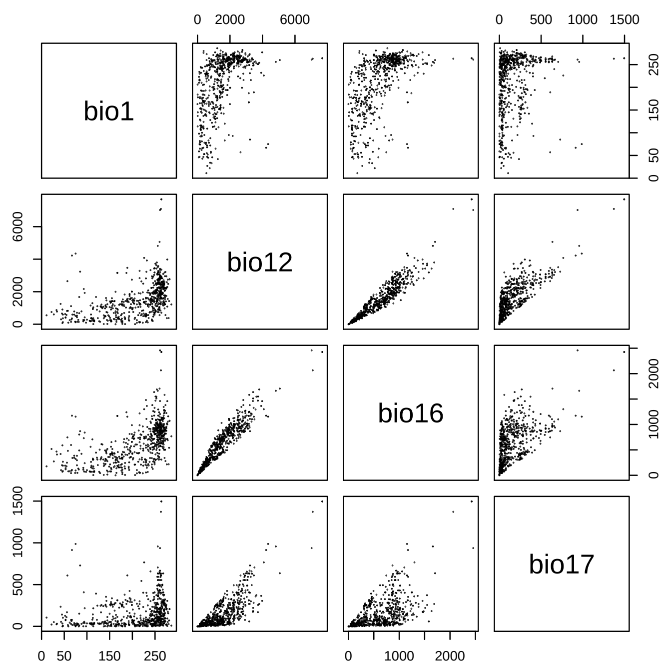
```
    pb bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8 biome
611  0  223   436   182    47  376   57  319  270    13
612  0  269   627   416     6  325  210  115  271    13
613  0  270  2563   982   254  329  223  106  265     1
614  0  256   510   276     4  339  168  171  263    13
615  0  272  1792  1013    22  323  226   97  267     1
616  0  124  1080   296   252  293  -42  334  223     5
```

```
> summary(sdmdata)
```

```
       pb              bio1            bio12            bio16       
 Min.   :0.0000   Min.   : -7.0   Min.   :   3.0   Min.   :   2.0  
 1st Qu.:0.0000   1st Qu.:181.8   1st Qu.: 842.8   1st Qu.: 354.5  
 Median :0.0000   Median :241.5   Median :1466.0   Median : 621.5  
 Mean   :0.1883   Mean   :214.0   Mean   :1581.1   Mean   : 643.1  
 3rd Qu.:0.0000   3rd Qu.:260.0   3rd Qu.:2212.8   3rd Qu.: 900.0  
 Max.   :1.0000   Max.   :285.0   Max.   :7682.0   Max.   :2458.0  

     bio17            bio5            bio6              bio7      
 Min.   :   0.0   Min.   : 77.0   Min.   :-191.00   Min.   : 60.0  
 1st Qu.:  36.0   1st Qu.:302.0   1st Qu.:  52.75   1st Qu.:118.0  
 Median : 110.0   Median :320.0   Median : 155.00   Median :164.5  
 Mean   : 162.5   Mean   :308.7   Mean   : 118.27   Mean   :190.4  
 3rd Qu.: 232.2   3rd Qu.:333.0   3rd Qu.: 200.00   3rd Qu.:239.5  
 Max.   :1496.0   Max.   :409.0   Max.   : 242.00   Max.   :447.0  
```

```
      bio8           biome    
 Min.   :-35.0   1      :285  
 1st Qu.:216.8   13     : 74  
 Median :251.0   7      : 71  
 Mean   :224.3   2      : 41  
 3rd Qu.:263.0   8      : 37  
 Max.   :322.0   (Other):107  
                 NA's   :  1  
```

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



4.3 Variable selection

Variable selection is obviously important, particularly if the objective of a study is explanation. See, e.g., Austin and Smith (1987), Austin (2002). The early applications of species modeling tended to focus on explanation (Elith and Leathwick 2009). Nowadays, the objective of SDM tends to be prediction, in which case variable selection might be less important (as long as there are enough variables with different spatial patterns); but this is an area that needs further research.

(to do: checking for correlations between predictors. How to choose a subset from a potentially large candidate set. Where to start (with the ecology of the species, rather than with the first data that can be found on the web!). Something about grid cell sizes, its relevance to the data content of the raster, what to do if you have some fine scale and some coarser scale data, how to deal with categorical variables with heaps of classes. The idea of presenting a new version of a variable that is more ecologically relevant to the species (e.g., looking at forest within x km of a grid cell, for a species that might have a home range of x) - using focal stats in raster).

Part II

Model fitting, prediction, and evaluation

Chapter 5

Model fitting

Model fitting is quite similar accross 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 the sections on specific modeling methods.

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.

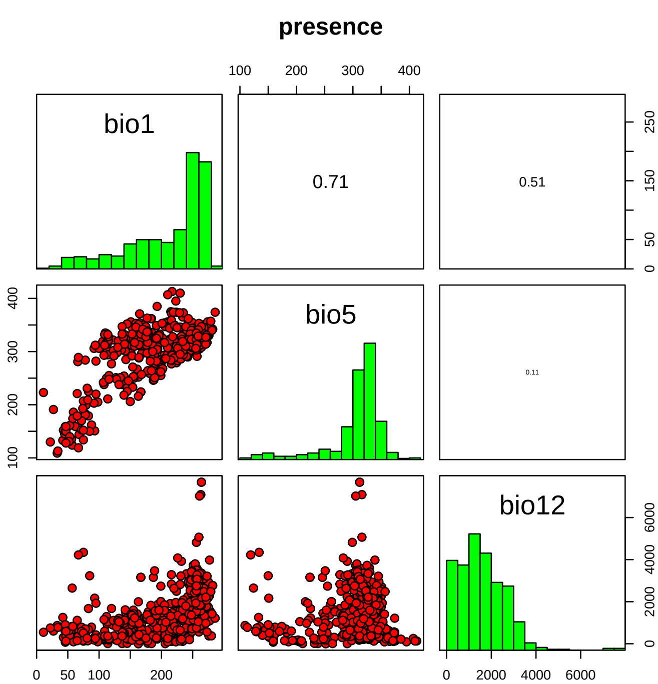
```
> m1 = glm(pb ~ bio1 + bio5 + bio12, data=sdmdata)
> class(m1)

[1] "glm" "lm"

> summary(m1)

Call:
glm(formula = pb ~ bio1 + bio5 + bio12, data = sdmdata)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.57793  -0.23632  -0.08191   0.08083   0.91252

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.382e-02  9.745e-02   0.757 0.449043
bio1         1.409e-03  4.042e-04   3.485 0.000528 ***
bio5        -1.285e-03  4.530e-04  -2.836 0.004722 **
bio12        1.313e-04  1.697e-05   7.735 4.27e-14 ***
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.1196562)

    Null deviance: 94.156  on 615  degrees of freedom
Residual deviance: 73.230  on 612  degrees of freedom
AIC: 446.27

Number of Fisher Scoring iterations: 2

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

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

Coefficients:
(Intercept)         bio1        bio12        bio16        bio17         bio5
  0.2155995   -0.0017383    0.0003716   -0.0003891   -0.0007653    0.0034685
       bio6         bio7         bio8       biome2       biome3       biome4
 -0.0024517   -0.0037371    0.0004697   -0.0974776   -0.1080345   -0.1287303
     biome5       biome7       biome8       biome9      biome10      biome12
 -0.0575800   -0.2338435   -0.0304285   -0.0331556   -0.0670728   -0.0355746
    biome13      biome14
  0.0261338   -0.1139393

Degrees of Freedom: 615 Total (i.e. Null);  596 Residual
Null Deviance:          94.16
Residual Deviance: 67.33        AIC: 426.6
```

Models implemented in dismo do not use a formula (and most models only take presence points). For example:

```
> bc = bioclim(sdmdata[,c('bio1', 'bio5', 'bio12')])
> class(bc)

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

> bc

class    : Bioclim

variables: bio1 bio5 bio12
```

```
presence points: 616
   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 illusrated in the example below where we make predictions with model object 'm1' for three records with values for variables bio1, bio5 and bio12 (the variables used in the example above to create object m1)

```
> 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
0.1318607 0.3474437 0.2061776
```

```
> predict(bc, pd)
```

```
[1] 0.000000000 0.006493506 0.381493506
```

Chapter 7

Model evaluation

Traditional measures of fit used in regression, such as r^2 and *p*-values have little place in species distribution modeling. For some methods these metrics do not apply. But even if they do, they should normally not be used as all the classic assumptions on which they are based (independence of data, normality of distributions) are typically strongly violated. In stead, 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), perhaps depending on the goal of the study. Many measueres are 'threshold dependent'. That means that a threshold must be set first (e.g. 0.5). 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. Cohen's *kappa* is an example of a threshold dependent model evaluation statistic.

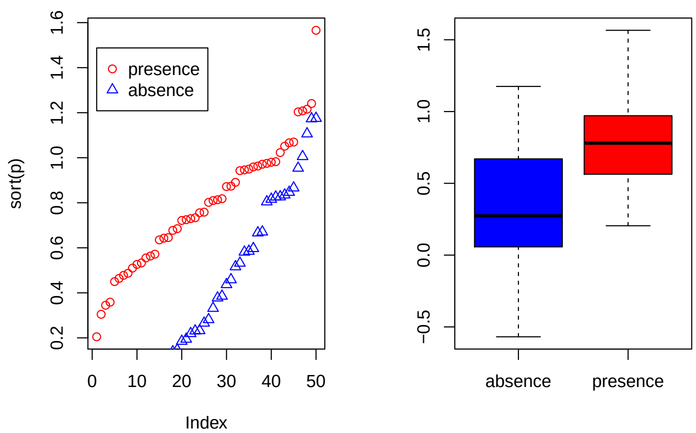
Much used statistics that are threshold independent are the correlation coefficient and the Area Under the Receiver Operator Curve (AUROC, generally further abbreviated to AUC). AUC is a measure of rank-correlation. If it is high, it indicates that high predicted scores tend to be areas of known presence and locations with lower model prediction scores tend to areas where the species is known to be absent (or a random point). An AUC score of 0.5 means that the model is as good as a random guess.

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

Create two variables with random normally distributed values and plot them:

```
> 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'))
```



The two variables clearly have different distributions, and the values for 'presence' tend to be higher than for 'absence'. Below we compute the correlation coeficient and the AUC:

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

      cor
0.5165112

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

[1] 0.7884
```

This is how you can computing these, and other statistics, with the dismo package (and see the ROCR package for similar functionality):

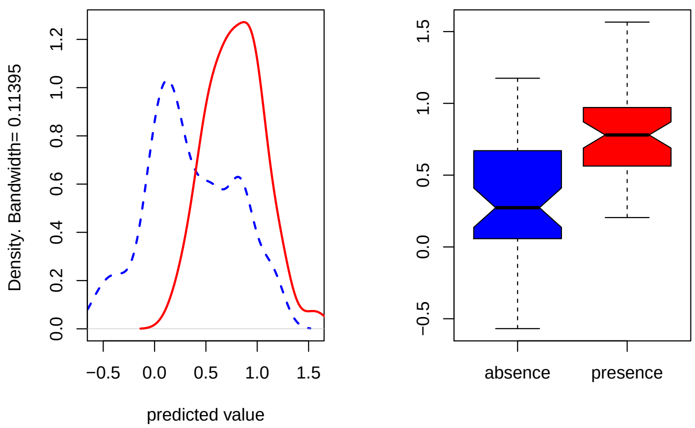
```
> e = evaluate(p=p, a=a)
> class(e)
```

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

```
> e
```

```
class              : ModelEvaluation
n presences        : 50
n absences         : 50
AUC                : 0.7884
cor                : 0.5165112
TPR+TNR threshold: 0.438
```

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



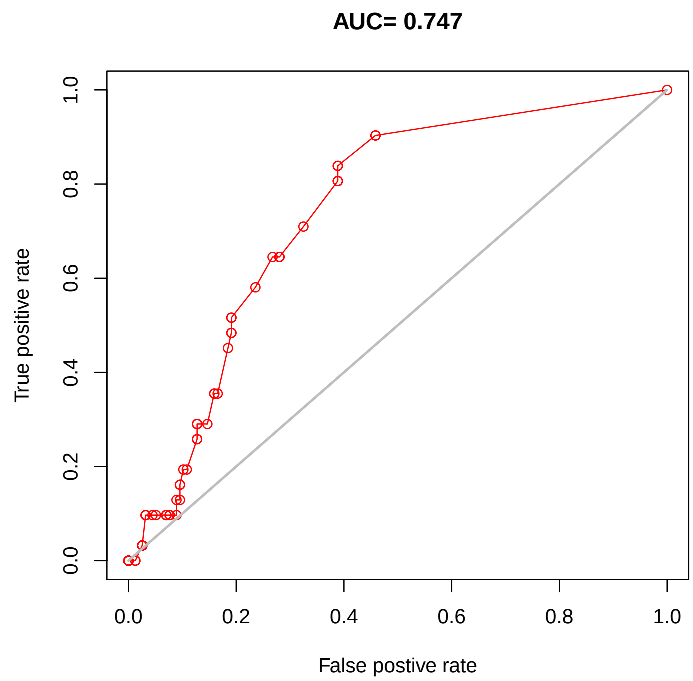
Now back to some real data, presence-only in this case. We'll divide the data in two random sets, one for training a Bioclim model, and one for evaluating the model.

```
> rand <- round(0.75 * runif(nrow(sdmdata)))
> traindata <- sdmdata[rand==0,]
> traindata <- traindata[traindata[,1] == 1, 2:9]
> testdata <- sdmdata[rand==1,]
> bc <- bioclim(traindata)
> e <- evaluate(testdata[testdata==1,], testdata[testdata==0,], bc)
> e
```

```
class              : ModelEvaluation
n presences        : 41
```

```
n absences       : 168
AUC              : 0.7576945
cor              : 0.181198
TPR+TNR threshold: 0.013
```

```
> plot(e, 'ROC')
```



Chapter 8

Data partitioning

The `kfold` function facilitates data partitioning. It creates a vector that assinges 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] 2 4 5 3 2 1 4 1 4 2

> unique(group)

[1] 2 4 5 3 1
```

Now we can fit and test our model five times. In each run, the records corresponding to one of the five group 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] 0.7240000 0.7972609 0.7592500 0.8306957 0.7676087

> mean(auc)

[1] 0.775763

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

[1] 0.022 0.022 0.040 0.060 0.040
```

Part III

Modeling methods

Chapter 9

Types of algorithms & data used in examples

A large number of algorithms have 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. Below we discuss examples of these different types of models.

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'll use the *Bradypus* data for presence of a species. Lets make a training and a testing set.

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

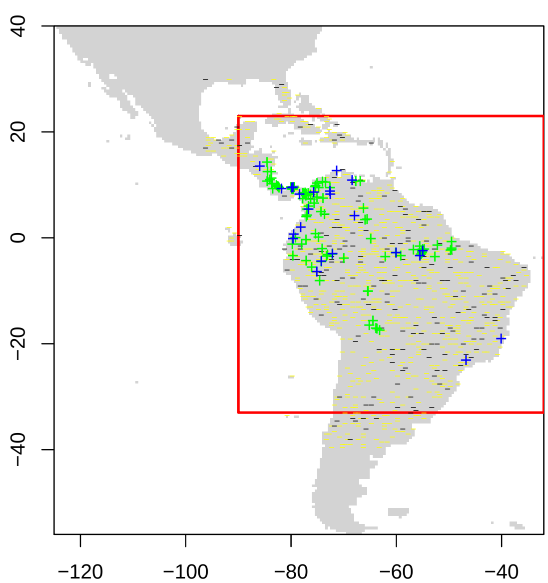
To speed up processing, let's restrict the predictions to a more restricted area (defined by a rectangular extent):

```
> 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 15% 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')
```



Chapter 10

Profile methods

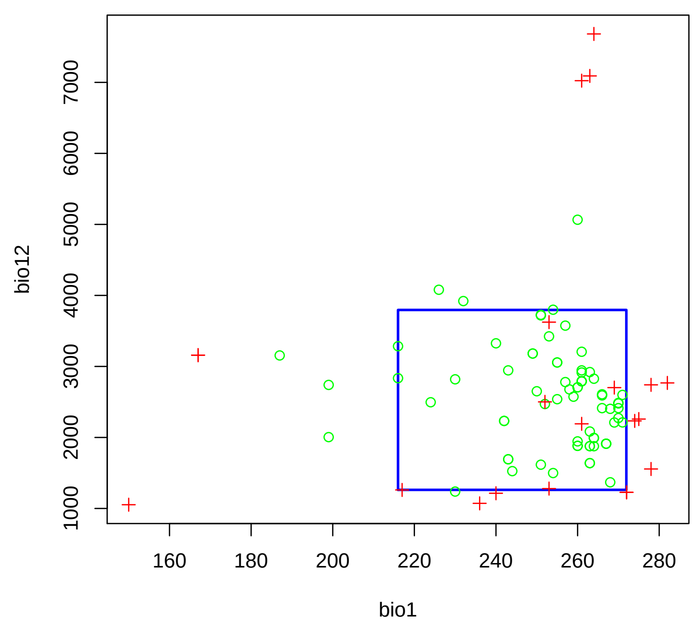
The three methods described here, Bioclim, Domain, and Mahal. These methods are implemented in the dismo package. The procedures to use these models is therefore the same for all three.

10.1 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 using an approach like Liebig's law of the minimum). This value is substracted from 1 and then mutliplied 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 distributon 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 assinged 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 simly using the predictors, and the occurrence points.

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



And evaluate it in a similar way, by providing presenced and background (absence) points, and a RasterStack:

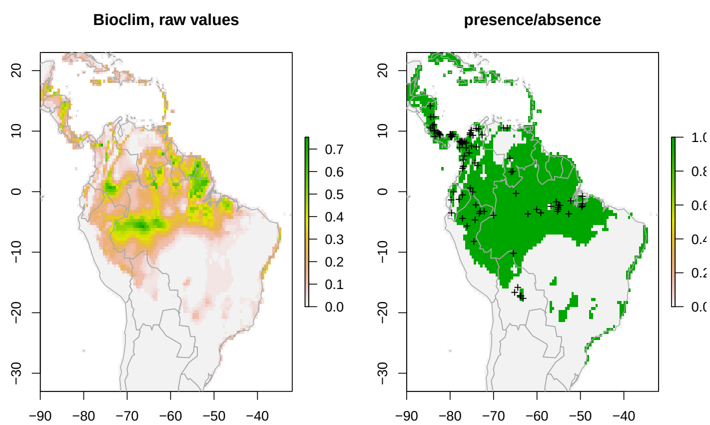
```
> e <- evaluate(pres_test, backg_test, bc, pred_nf)
> e

class              : ModelEvaluation
n presences        : 23
n absences         : 200
AUC                : 0.6572826
cor                : 0.09575775
TPR+TNR threshold: 0.04
```

And 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)
projection  : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
values      : in memory
min value   : 0
max value   : 0.7526882
```

```
> par(mfrow=c(1,2))
> plot(pb, main='Bioclim, raw values')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> threshold <- e@t[which.max(e@TPR + e@TNR)]
> plot(pb > threshold, 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 effecient 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.).

10.2 Domain

The Domain algorithm (Carpenter et al. 1993) that 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 beteen 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 Domain algorithm assigns to a place the distance to the closest known occurrence (in environmental space).

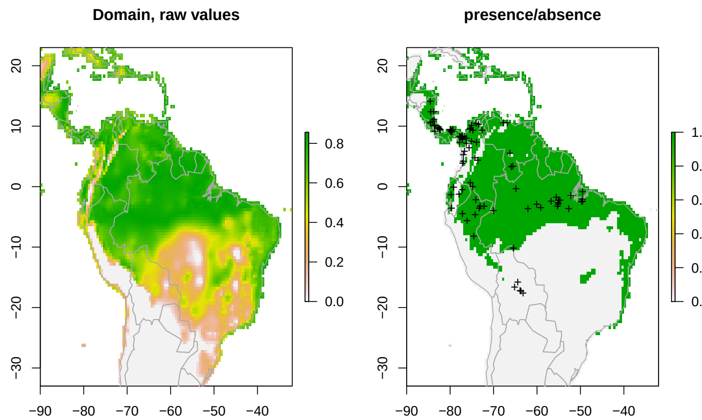
To integrate over environmental variables, the distance to any of the variables is used. This distance is substracted from one, and (in this R implementation) values below zero are truncated so that the scores are between 0 (low) and 1 (high).

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              : 0.6991304
cor              : 0.2091259
TPR+TNR threshold: 0.619
```

```
> 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')
> threshold <- e@t[which.max(e@TPR + e@TNR)]
> plot(pd > threshold, main='presence/absence')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> points(pres_train, pch='+')
```



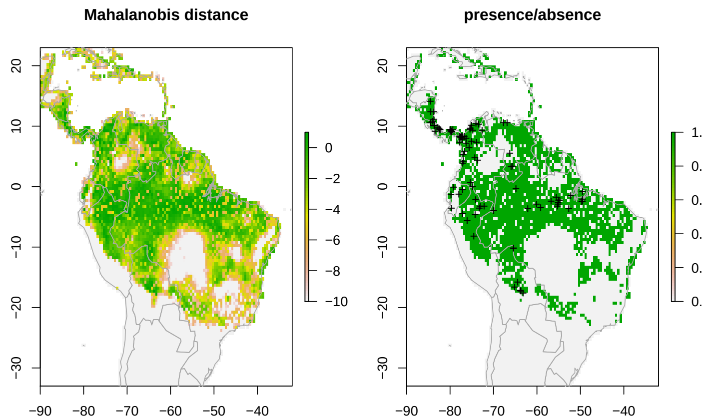
10.3 Mahalanobis

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              : 0.7096739
cor              : 0.1176877
TPR+TNR threshold: -3.199
```

```
> 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')
> threshold <- e@t[which.max(e@TPR + e@TNR)]
> plot(pm > threshold, main='presence/absence')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> points(pres_train, pch='+')
```


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

Chapter 11

Regression models

The remaining models need to be fit presence textifand 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)
```

11.1 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 (mulitple) 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 specfied with the 'family' argument. Examples are:

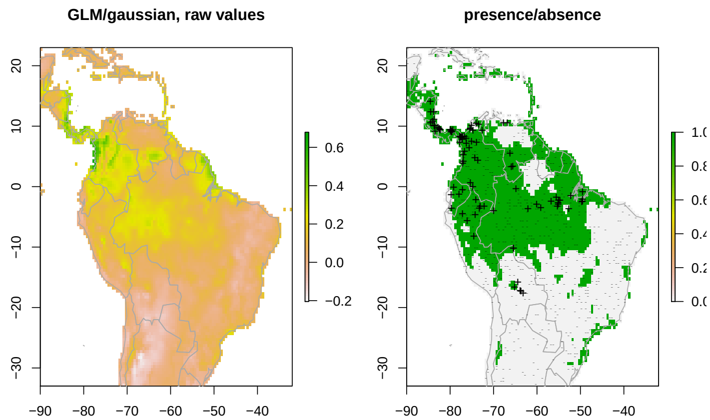
`family = binomial(link = "logit")`

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

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

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

```
> gm1 <- glm(pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17,
+                          family = binomial(link = "logit"), data=envtrain)
> gm2 <- glm(pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17,
+                          family = gaussian(link = "identity"), data=envtrain)
> e1 = evaluate(testpres, testbackg, gm1)
> e2 = evaluate(testpres, testbackg, gm2)
> e1

class            : ModelEvaluation
n presences      : 23
n absences       : 200
AUC              : 0.7892391
cor              : 0.2638338
TPR+TNR threshold: -2.919

> e2

class            : ModelEvaluation
n presences      : 23
n absences       : 200
AUC              : 0.7433696
cor              : 0.2979
TPR+TNR threshold: 0.091

> 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')
> threshold <- e2@t[which.max(e@TPR + e@TNR)]
> plot(pg > threshold, main='presence/absence')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> points(pres_train, pch='+')
> points(backg_train, pch='-', cex=0.25)
```



11.2 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. The 'grasp' package implements species distribution modeling with gam (Lehman et al., 2002).

Chapter 12

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 Regressin 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 accesible 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, 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.

12.1 Maxent

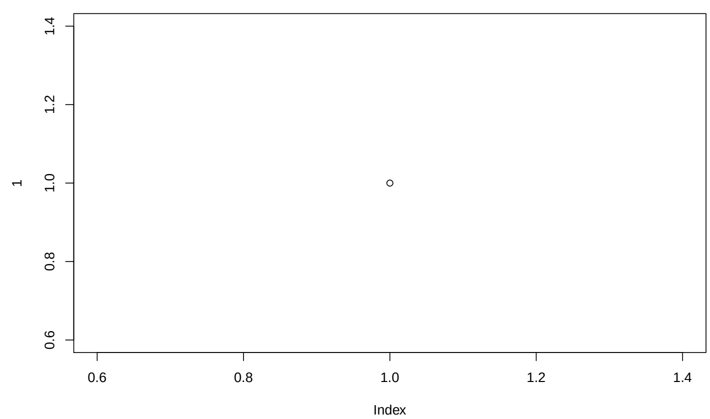
The Maxent (Maximum Entropy) species distribution model (Phillips et al., 2004, 2006) is a stand alone Java program. Dismo has a function 'maxent' that communicates with this program. To use it you must first download the program from `http://www.cs.princeton.edu/~schapire/maxent/`. Put the file 'maxent.jar' 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`) can not 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 fit it like the other methods such as glm. Note, however, that in that case you cannot use the formula notation.

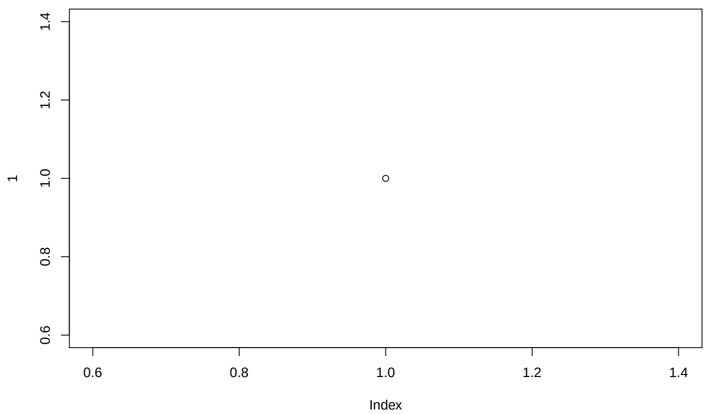
```
> jar <- paste(system.file(package="dismo"), "/java/maxent.jar", sep='')
> if (file.exists(jar)) {
```

```
+         xm <- maxent(predictors, pres_train, factors='biome')
+         plot(xm)
+ } else {
+     cat('cannot run this example because maxent is not available on this system')
+         plot(1)
+ }
```

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




```
> 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')
+         threshold <- e@t[which.max(e@TPR + e@TNR)]
+         plot(px > threshold, main='presence/absence')
+         plot(wrld_simpl, add=TRUE, border='dark grey')
+         points(pres_train, pch='+')
+ } else {
+         plot(1)
+ }
```



12.2 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. The functions have been slightly adjusted and incorporated into the 'dismo' package. These funcitons extend the funcitons 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')`

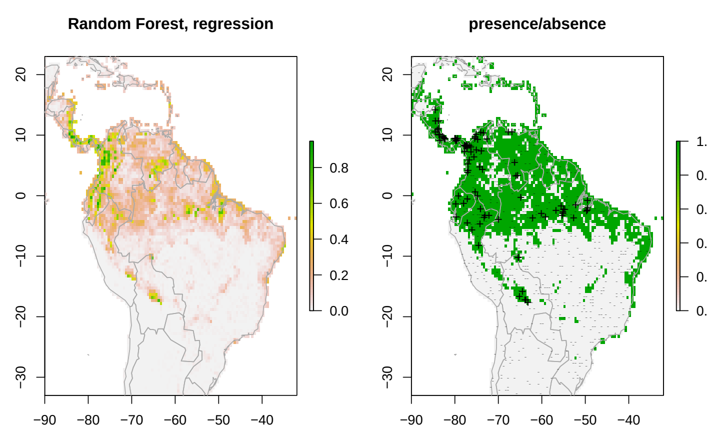
12.3 Random Forest

The Random Forest (Breiman, 2001b) method is an extention 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 seperate argumetns, 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 exactely 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))
> e = evaluate(testpres, testbackg, rf1)
> e
```

```
class          : ModelEvaluation
n presences    : 23
n absences     : 200
AUC            : 0.7690217
cor            : 0.3846472
TPR+TNR threshold: 0.18
```

```
> 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')
> threshold <- e@t[which.max(e@TPR + e@TNR)]
> plot(pr > threshold, main='presence/absence')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> points(pres_train, pch='+')
> points(backg_train, pch='-', cex=0.25)
```



12.4 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 pakcage '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)

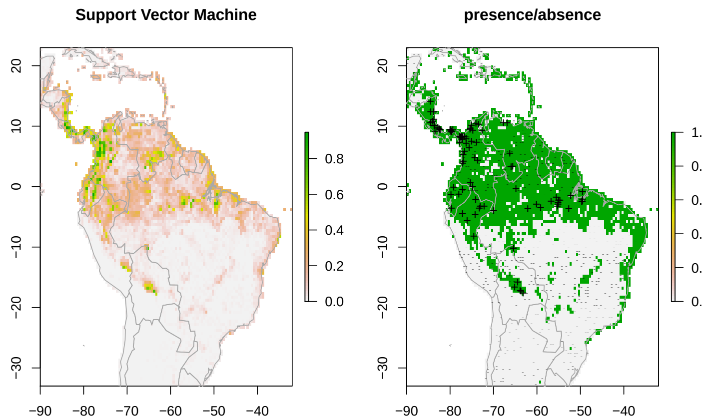
```
> library(kernlab)
> svm <- ksvm(pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17, data=envtrain)
```

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

```
> e = evaluate(testpres, testbackg, svm)
> e
```

```
class             : ModelEvaluation
n presences       : 23
n absences        : 200
AUC               : 0.6153261
cor               : 0.2003148
TPR+TNR threshold: 0.027
```

```
> ps <- predict(predictors, rf1, ext=ext)
> par(mfrow=c(1,2))
> plot(ps, main='Support Vector Machine')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> threshold <- e@t[which.max(e@TPR + e@TNR)]
> plot(ps > threshold, main='presence/absence')
> plot(wrld_simpl, add=TRUE, border='dark grey')
> points(pres_train, pch='+')
> points(backg_train, pch='-', cex=0.25)
```



The remainder of this document is to be completed.

12.5 Other methods

Neural networks, ...

Chapter 13

More...

Part IV

Multi-species models

13.1 mars

13.2 gdm

Part V

Model transfer in space and time

Chapter 14

Transfer in space

Chapter 15

Transfer in time: climate change

Part VI

Model averaging, uncertainty

See the BIOMOD for on multi-model inference.
Dealing with uncertainty

Part VII

Geographic models

The 'geographic models' described here are not commonly used in species distribution modeling. They are an attempt to formalize methods to draw 'expert range maps'. They can also be interpreted as null-models. To be completed.

Chapter 16

Geographic models (presence-only)

16.1 Distance

16.2 Convex hulls

16.3 Circles

Chapter 17

Geographic models (presence-absence)

17.1 Inverse distance

17.2 Voronoi hulls

Part VIII

References

Austin MP, 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.

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.

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. 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., 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., 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., Thomas C. Edwards, Jr, and Trevor 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., 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`

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/`

Lehmann, A., J. McC. Overton and J.R. Leathwick, 2002. GRASP: Generalized Regression Analysis and Spatial Predictions. Ecological Modelling 157: 189-207.

Mahalanobis, P.C., 1936. On the generalised distance in statistics. Proceedings of the National Institute of Sciences of India 2: 49-55.

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

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.

Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.

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.