A very brief introduction to species distribution models in R

Jeff Oliver

26 February, 2018

Outline:

- What they are (introductory stuff + description)
- What do we need?
  - Data
    - Occurrence
    - Bioclim
  - Dependencies
- Preparing data?
- Quick plot of points on map
- Pseudo-absence points
- Models
  - Quick glance at loadings?
- Maps
  - Do just probability map, without any pseudo-absence points
  - Do threshold map, with pseudo-absence points

Predicting ranges' of species from latitude and longitude coordinates has become increasingly easier with a suite of R packages. This introductory tutorial will show you how to turn your coordinate data into a range map.

Learning objectives

1. Install packages for species distribution modeling
2. Run species distribution models using `bioclim` approach
3. Visualize model predictions on a map

[DESCRIPTION OR MOTIVATION; 2-4 sentences that would be used for an announcement]

An introduction to predicting species' geographic ranges based on occurrence and climate data. We'll be using publicly available data to build models of species distributions and to generate distribution maps.

Getting started

Before we do anything, we will need to set up our workspace, download example data, and install additional packages that are necessary to run the models and visualize their output.

Workspace organization

So, to start, create a pair of folders in your workspace:

```
dir.create(path = "data")
dir.create(path = "output")
```

It is good practice to keep input (i.e. the data) and output separate. Furthermore, any work that ends up in the `output` folder should be completely disposable. That is, the combination of data and the code we write should allow us (or anyone else, for that matter) to reproduce any output.

Example data

The data we are working with are observations of the Saguaro, *Carnegiea gigantea*. We are using a subset of records available from GBIF, the Global Biodiversity Information Facility. You can download the data from ; save it in the `data` folder that you created in the step above.

Install additional R packages

Next, there are *five* additional R packages that will need to be installed:

- dismo
- maptools
- rgdal
- raster
- sp

To install these, run:

```
install.packages("dismo")
install.packages("maptools")
install.packages("rgdal")
install.packages("raster")
install.packages("sp")
```

Ingredient list

The basic idea behind species distribution models is to take two sources of information to model the conditions in which a species is expected to occur. The two sources of information are:

1. Occurrence data: these are usually latitude and longitude geographic coordinates where the species of interest has been observed. These are known as 'presence' data. Some models also make use of 'absence' data, which are geographic coordinates of locations where the species is known to *not* occur. Absence data are a bit harder to come by, but are required by some modeling approaches. For this lesson, we will use the occurrence data of the Saguaro that you downloaded earlier.
2. Environmental data: these are descriptors of the environment, and can include abiotic measurements of temperature and precipitation as well as biotic factors, such as the presence or absence of other species (like predators, competitors, or food sources). In this lesson we will focus on the 19 abiotic variables available from WorldClim. Rather than downloading the data from WorldClim, we'll use functions from the `dismo` package to download these data (see below).

Data and quality control

We'll start our script by loading those five libraries we need. And of course adding a little bit of information at the very top of our script that says what the script does and who is responsible!

```
# Species distribution modeling for Saguaro
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2018-02-27

library("sp")
library("raster")
library("maptools")
library("rgdal")
library("dismo")
```

There is a good chance you might have seen some red messages print out to the screen, especially when loading the `maptools` or `rgdal` libraries. This is normal, and as long as none of the messages include "ERROR", you can just hum right through those messages. If loading the libraries *does* result in an ERROR message, check to see that the libraries were installed properly.

Now that we have those packages loaded, we can download the bioclimatic variable data with the `getData` function:

```
bioclim.data <- getData(name = "worldclim",
                        var = "bio",
                        res = 2.5,
                        path = "data/")
```

We're giving `getData` four critical pieces of information:

1. `name = "worldclim"`: This indicates the name of the data set we would like to download
2. `var = "bio"`: This tells `getData` that we want to download all 19 of the bioclimatic variables, rather than individual temperature or precipitation measurements
3. `res = 2.5`: This is the resolution of the data we want to download; in this case, it is 2.5 minutes of a degree. For other resolutions, you can check the documentation by typing `?getData` into the console.
4. `path = "data/"`: Finally, this sets the location to which the files are downloaded. In our case, it is the `data` folder we created at the beginning.

Note also that after the files are downloaded to the `data` folder, the are read into memory and stored in the variable called `bioclim.data`

```
# Read in saguaro observations
obs.data <- read.csv(file = "data/Carnegiea-gigantea-GBIF.csv")

# Check the data to make sure it loaded correctly
summary(obs.data)
```

```
##      gbifid             latitude       longitude
##  Min.   :2.021e+08   Min.   :26.78   Min.   :-114.0
##  1st Qu.:1.453e+09   1st Qu.:32.17   1st Qu.:-111.4
##  Median :1.571e+09   Median :32.28   Median :-111.1
##  Mean   :1.567e+09   Mean   :32.16   Mean   :-111.3
##  3rd Qu.:1.677e+09   3rd Qu.:32.38   3rd Qu.:-111.0
##  Max.   :1.806e+09   Max.   :34.80   Max.   :-109.3
##                      NA's   :3       NA's   :3
```

Notice that there are three `NA` values in the `latitude` and `longitude` columns. Those records will not be of

any use to us, so we can remove them from our data frame:

```
# Notice NAs - drop them before proceeding
obs.data <- obs.data[!is.na(obs.data$latitude), ]

# Make sure those NA's went away
summary(obs.data)
```

```
##      gbifid              latitude       longitude
##  Min.   :8.910e+08   Min.   :26.78   Min.   :-114.0
##  1st Qu.:1.453e+09   1st Qu.:32.17   1st Qu.:-111.4
##  Median :1.571e+09   Median :32.28   Median :-111.1
##  Mean   :1.575e+09   Mean   :32.16   Mean   :-111.3
##  3rd Qu.:1.677e+09   3rd Qu.:32.38   3rd Qu.:-111.0
##  Max.   :1.806e+09   Max.   :34.80   Max.   :-109.3
```

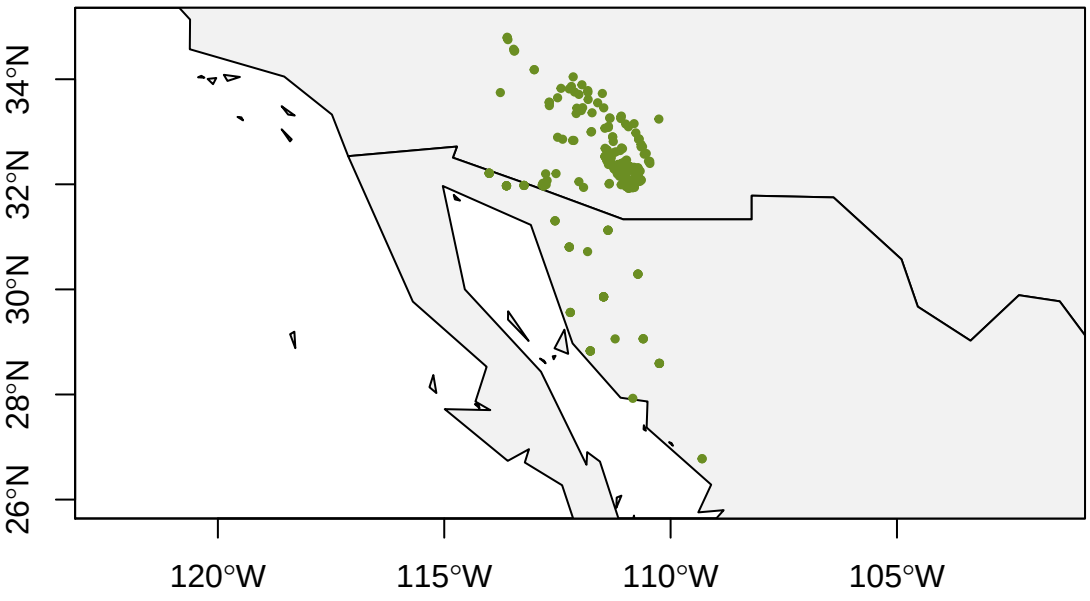
When we look at the `obs.data` data frame now there are no `NA` values, so we are ready to proceed.

To make species distribution modeling more streamlined, it is useful to have an idea of how widely our species is geographically distributed. We are going to find general latitudinal and longitudinal boundaries and store this information for later use:

```
# Determine geographic extent of our data
max.lat = ceiling(max(obs.data$latitude))
min.lat = floor(min(obs.data$latitude))
max.lon = ceiling(max(obs.data$longitude))
min.lon = floor(min(obs.data$longitude))
geographic.extent <- extent(x = c(min.lon, max.lon, min.lat, max.lat))
```

Before we do any modeling, it is also a good idea to run a reality check on your occurrence data by plotting the points on a map.

```
# Load the data to use for our base map
data(wrld_simpl)

# Plot the base map
plot(wrld_simpl,
     xlim = c(min.lon, max.lon),
     ylim = c(min.lat, max.lat),
     axes = TRUE,
     col = "grey95")

# Add the points for individual observation
points(x = obs.data$longitude,
       y = obs.data$latitude,
       col = "olivedrab",
       pch = 20,
       cex = 0.75)
# And draw a little box around the graph
box()
```



Building a model and visualizing results

Now that our occurrence data look OK, we can use the bioclimatic variables to create a model. The first thing we want to do though is limit our consideration to a reasonable geographic area. That is, for our purposes we are not looking to model Saguaro habitat suitability *globally*, but rather to the general southwest region. So we can restrict the biolimatic variable data to the geographic extent of our occurrence data:

```
# Crop bioclim data to geographic extent of saguaro
bioclim.data <- crop(x = bioclim.data, y = geographic.extent)

# Build species distribution model
bc.model <- bioclim(x = bioclim.data, p = obs.data)
```

```
## Error in .xyValues(x, as.matrix(y), ...): xy should have 2 columns only.
## Found these dimensions: 400, 3
```

Uh oh. That's not good. It looks like the data we passed to `bioclim` is not in the right format. The clue comes in the second line of the error message: `## Found these dimensions: 400, 3`. This is referring to the `obs.data` data frame, which does indeed have 400 rows and three columns. From the documentation from `bioclim` (see for yourself via `?bioclim` in the console):

> **Usage**
> bioclim(x, p, …)
> **Arguments**
> x Raster* object or matrix
> p two column matrix or SpatialPoints* object
> … Additional arguments

So whatever we pass to `p` should only have **two** columns. Let's modify the `obs.data` so it only has two columns. The first column is the GBIF identifier, which we will not need, so we drop it using the negation operator (i.e. the minus sign). Then we can run the species distribution model.

```
# Drop unused column
obs.data <- obs.data[, -1]
```

```
# Build species distribution model
bc.model <- bioclim(x = bioclim.data, p = obs.data)
```

```
## Error in bioclim(data.frame(m), ...): insufficient records
```

What the…? OK, this error message is tougher to figure out. But let's consider what our `obs.data` data frame looks like now:

```
head(obs.data)
```

```
##   latitude longitude
## 1 32.33556 -110.8980
## 2 32.28267 -110.9028
## 3 30.29105 -110.7213
## 4 32.05413 -110.6837
## 5 32.25111 -110.7169
## 6 32.19404 -111.0198
```

The first column is latitude and the second column is longitude, which seems fine. That is, until we think about how R generally deals with coordinates. When we plot something, we generally use syntax like this:

```
plot(x, y)
```

The thing to note is that the first argument we pass is data for the **x-axis** and the second argument is for the **y-axis**. The `bioclim` function is looking for data in the *same order*. That is, it looks at whatever we passed to `p` and assumes the first column is for the x-axis and the second column is for the y-axis. But our data is in the opposite order: the first column is *latitude*, essentially the y-axis data, and our second column is longitude, corresponding to x-axis data. So we need to reverse the column order before we pass `obs.data` to `bioclim`:

```
# Reverse order of columns
obs.data <- obs.data[, c(2, 1)]

# Build species distribution model
bc.model <- bioclim(x = bioclim.data, p = obs.data)
```

Woo-hoo! No errors here (hopefully).

There's one more step we need to take before we plot the model on a map. We need to generate an object that has the model's probability of occurrence for saguaros. We use the `predict` model from the `dismo` package:

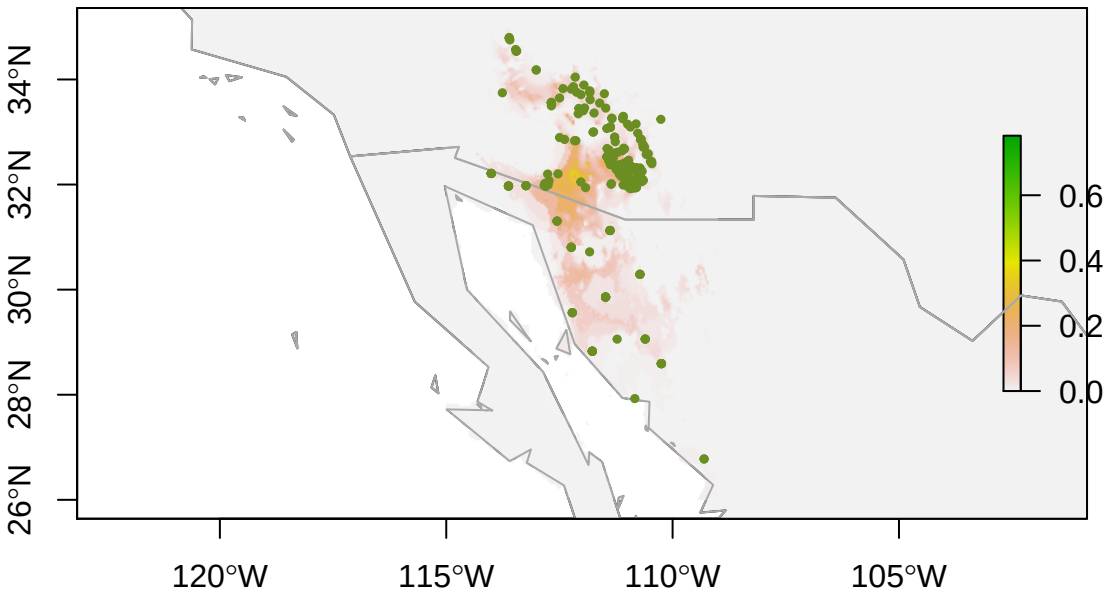
```
# Predict presence from model
predict.presence <- dismo::predict(object = bc.model, x = bioclim.data, ext = geographic.extent)
```

You might be wondering about why we use `dismo::predict` rather than just `predict`. Not surprisingly, different packages sometimes use the same function name to perform very different operations. In the case of `predict`, there are at least three packages loaded into memory that have a `predict` function: `dismo`, `sp`, and `stats`. Although we *probably* would have been fine just using `predict` (R would have use the `dismo` version), specifying the `dismo` version explicitly communicates this fact to anyone else reading the code. So, rather than leaving others (or your future self!) guessing, we can use the `dismo::predict` syntax.

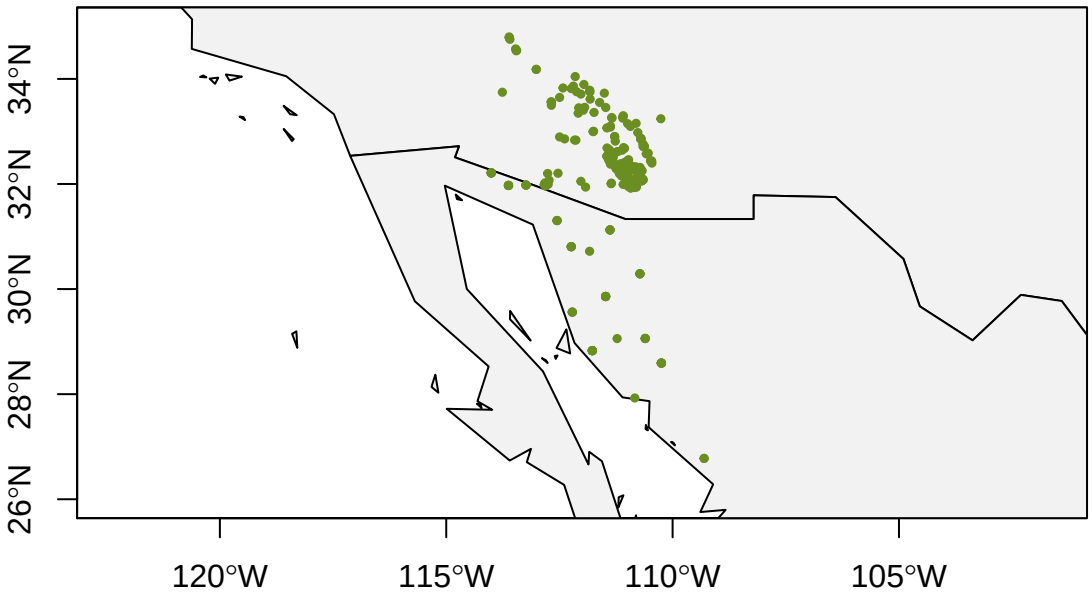
Enough! It's time to plot. We start as we did before, with a blank gray map, add the model, and if we feel like it, add the original observations as points.

```
# Plot base map
plot(wrld_simpl,
     xlim = c(min.lon, max.lon),
     ylim = c(min.lat, max.lat),
```

```
     axes = TRUE,
     col = "grey95")

# Add model probabilities
plot(predict.presence, add = TRUE)

# Redraw those country borders
plot(wrld_simpl, add = TRUE, border = "dark grey")

# Add original observations
points(obs.data$longitude, obs.data$latitude, col = "olivedrab", pch = 20, cex = 0.75)
box()
```



This plot shows the probability of occurrence of saguaros across the map. Note the values are all quite below 1.0; in fact, the maximum probability anywhere on the map is only 0.78, according to the model. However, we are pretty sure that saguaros are found across a pretty broad area of the Sonoran Desert - after all, we have the observations to prove that! If we want our map to better reflect this, we will need to re-run our analyses, but this time include some absence points, where saguaros are known to *not* occur. The problem is, we only have presence data for saguaros.



The pseudo-absence point

One common work around for coercing presence-only data for use with presence/absence approaches is to use pseudo-absence, or "background" points. While "pseudo-absence" sounds fancy, it really just means that one randomly samples points from a given geographic area and treats them like locations where the species of interest is absent. A great resource investigating the influence and best practices of pseudo-absence points is a study by Barbet-Massin *et al.* (2012) (see References below for full details).

For our purposes, we are going to create a set of background (aka pseudo-absence) points at random, with as many points as we have observations. We are going to use the bioclim data files for determining spatial resolution of the points, and restrict the sampling area to the general region of the observations of saguaros.

```
# Use the bioclim data files for sampling resolution
bil.files <- list.files(path = "data/wc2-5",
                        pattern = "*.bil$",
                        full.names = TRUE)

# We only need one file, so use the first one in the list of .bil files
mask <- raster(bil.files[1])

# Randomly sample points (same number as our observed points)
background <- randomPoints(mask = mask,      # Provides resolution of sampling points
                   n = nrow(obs.data),       # Number of random points
                   ext = geographic.extent, # Spatially restricts sampling
                   extf = 1.25)              # Expands sampling a little bit
```

Take a quick look at the `background` object we just created:

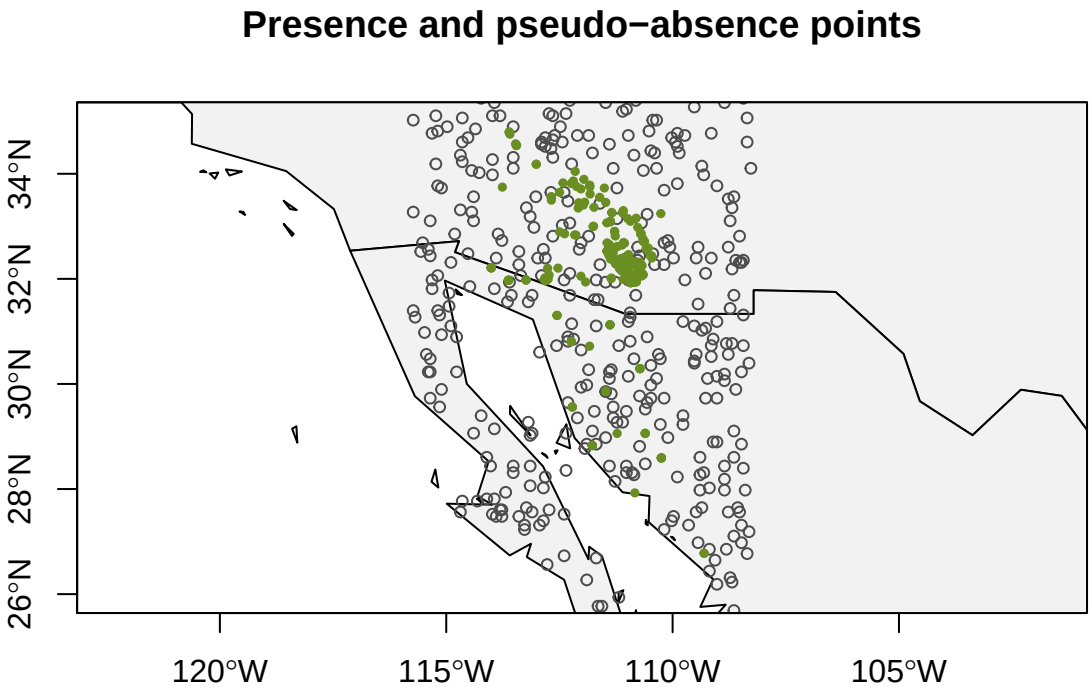
```
head(background)
```

```
##              x        y
## [1,] -111.7292 31.60417
## [2,] -110.2708 30.47917
```

```
## [3,] -110.1875 32.68750
## [4,] -110.7292 32.43750
## [5,] -109.2708 29.72917
## [6,] -111.3542 29.81250
```

We can also visualize them on a map, like we did for the observed points:

```
# Plot the base map
plot(wrld_simpl,
     xlim = c(min.lon, max.lon),
     ylim = c(min.lat, max.lat),
     axes = TRUE,
     col = "grey95",
     main = "Presence and pseudo-absence points")

# Add the background points
points(background, col = "grey30", pch = 1, cex = 0.75)

# Add the observations
points(x = obs.data$longitude,
       y = obs.data$latitude,
       col = "olivedrab",
       pch = 20,
       cex = 0.75)

box()
```



Now that we have our pseudo-absence points, we need to take one more step. Getting a more traditional-range-map-looking figure requires *post hoc* evaluation of the model. To do this evaluation, we are going to build the model using only *part* of our data (the **training** data), reserving a portion of the data for evaluation of the model after it is build (the **testing** data). We are going to reserve 20% of the data for testing, so we use the `kfold` function in the `dismo` package to evenly assign each observation to a random group.

```
# Arbitrarily assign group 1 as the testing data group
testing.group <- 1

# Create vector of group memberships
group.presence <- kfold(x = obs.data, k = 5) # kfold is in dismo package
```

Now pause for a minute and take a look at that `group.presence` vector we just created:

```
head(group.presence)
```

```
## [1] 4 1 2 4 4 3
```

```
# Should see even representation in each group
table(group.presence)
```

```
## group.presence
##  1  2  3  4  5
## 80 80 80 80 80
```

The output of `table` shows how many points have been assigned to each of the five groups.

We use the `group.presence` vector with the observed data to separate our observations into a training data set and a testing data set:

```
# Separate observations into training and testing groups
presence.train <- obs.data[group.presence != testing.group, ]
presence.test <- obs.data[group.presence == testing.group, ]

# Repeat the process for pseudo-absence points
group.background <- kfold(x = background, k = 5)
background.train <- background[group.background != testing.group, ]
background.test <- background[group.background == testing.group, ]
```

Training and testing the model

Now that we have (1) our pseudo-absence points and (2) separate training and testing data, we can re-build the model, evaluate its performance, and draw a more aesthetically pleasing map. We build the model with the `bioclim` function as before, but instead of using all the observations in `obs.data` we only use the training data stored in `presence.train`:

```
# Build a model using training data
bc.model <- bioclim(x = bioclim.data, p = presence.train)
```

We now take that model, and evaluate it using the observation data and the pseudo-absence points we reserved for model *testing*. We then use this test to establish a cutoff of occurrence probability to determine the boundaries of the saguaro range.

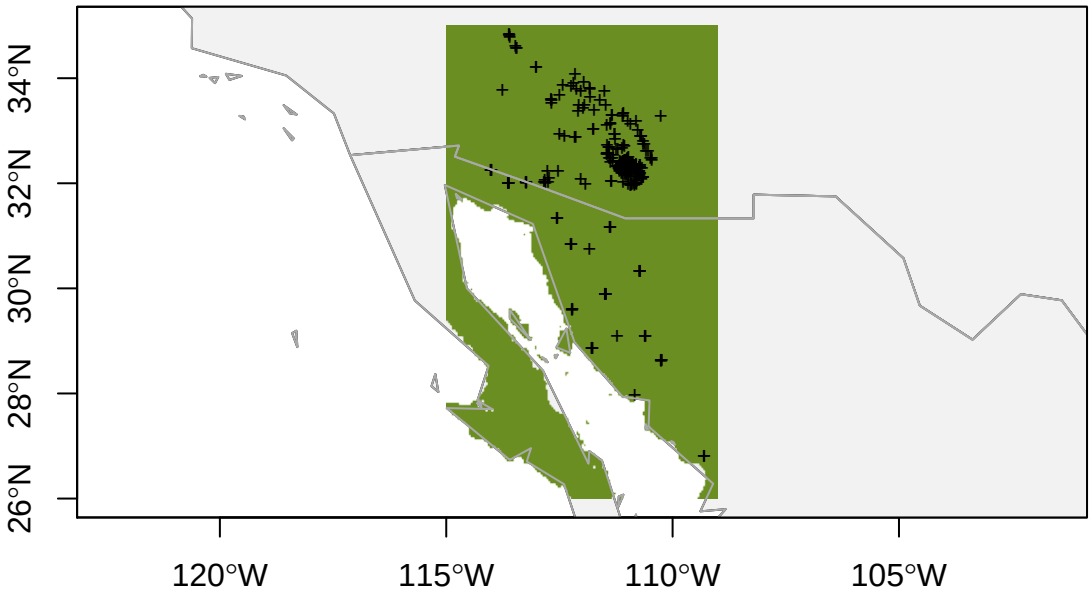
```
# Use testing data for model evaluation
bc.eval <- evaluate(p = presence.test,    # The presence testing data
                    a = background.test, # The absence testing data
                    model = bc.model,    # The model we are evaluating
                    x = bioclim.data)    # Climatic variables for use by model

# Determine minimum threshold for "presence"
bc.threshold <- threshold(x = bc.eval, stat = "spec_sens")
```

The `threshold` functions offers a number of means of determining the threshold cutoff through the `stat`

parameter. Here we chose `"spec_sens"`, which sets "the threshold at which the sum of the sensitivity (true positive rate) and specificity (true negative rate) is highest." For more information, check out the documentation for `threshold` (`?threshold`, remember?).

And *finally*, we can use that threshold to paint a map with the predicted range of the saguaro!

```
# Predict presence from model (same as previously, but with the update model)
predict.presence <- dismo::predict(object = bc.model,
                                   x = bioclim.data,
                                   ext = geographic.extent)

# Plot base map
plot(wrld_simpl,
     xlim = c(min.lon, max.lon),
     ylim = c(min.lat, max.lat),
     axes = TRUE,
     col = "grey95")

# Only plot areas where probability of occurrence is greater than the threshold
plot(predict.presence > bc.threshold,
     add = TRUE,
     legend = FALSE,
     col = "olivedrab")

# And add those observations
points(x = obs.data$longitude,
       y = obs.data$latitude,
       col = "black",
       pch = "+",
       cex = 0.75)

# Redraw those country borders
plot(wrld_simpl, add = TRUE, border = "dark grey")
box()
```



Hmmm…that doesn't look right. It plotted a large portion of the map green. Let's look at what we actually

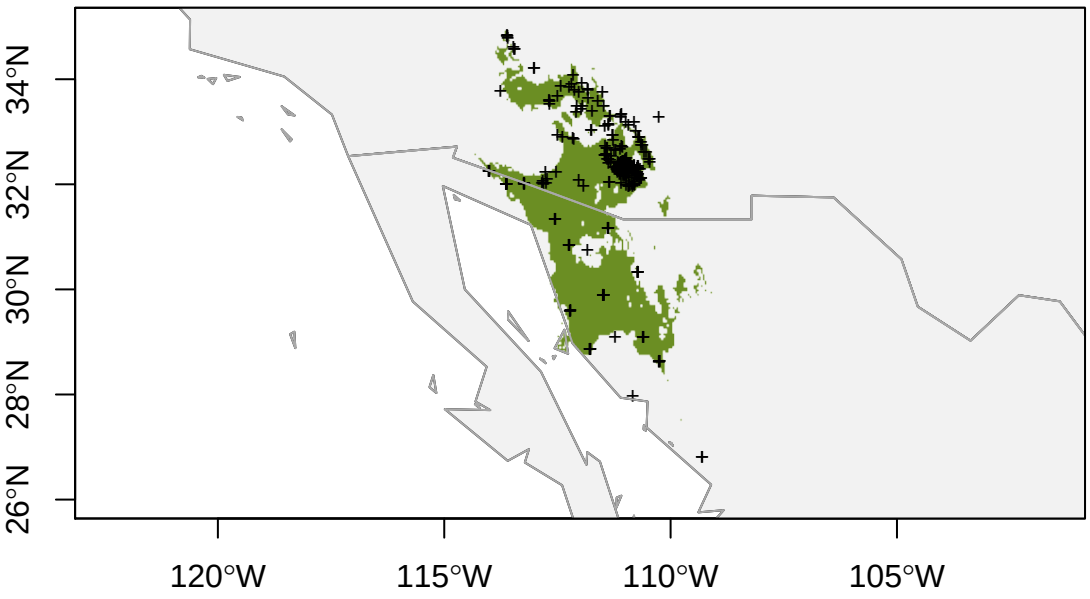
asked R to plot, that is, we plot the value of `predict.presence > bc.threshold`. So what is that?

```
predict.presence > bc.threshold
```

```
## class       : RasterLayer
## dimensions  : 216, 144, 31104  (nrow, ncol, ncell)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : -115, -109, 26, 35  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : layer
## values      : 0, 1  (min, max)
```

The comparison of these two rasters produces another raster with values of only 0 or 1: 0 where the comparison evaluates as `FALSE` (i.e., when the value in a grid cell of predict.presence is less than or equal to the value in the corresponding grid cell of bc.threshold) and 1 where the comparison evaluates at `TRUE`. Since there are two values in this comparison (the 0 and 1 in the `values` field), we need to update what we pass to the `col` parameter in our plot call. Instead of just passing a single value, we provide a color for 0 (`NA`) and a color for 1 (`"olivedrab"`):

```
# Predict presence from model (same as previously, but with the update model)
predict.presence <- dismo::predict(object = bc.model,
                                   x = bioclim.data,
                                   ext = geographic.extent)

# Plot base map
plot(wrld_simpl,
     xlim = c(min.lon, max.lon),
     ylim = c(min.lat, max.lat),
     axes = TRUE,
     col = "grey95")

# Only plot areas where probability of occurrence is greater than the threshold
plot(predict.presence > bc.threshold,
     add = TRUE,
     legend = FALSE,
     col = c(NA, "olivedrab"))

# And add those observations
points(x = obs.data$longitude,
       y = obs.data$latitude,
       col = "black",
       pch = "+",
       cex = 0.75)

# Redraw those country borders
plot(wrld_simpl, add = TRUE, border = "dark grey")
box()
```



Additional resources

- Vignette for `dismo` package
- Fast and flexible Bayesian species distribution modelling using Gaussian processes
- Species distribution models in R
- Run a range of species distribution models
- SDM polygons on a Google map
- R package 'maxnet' for functionality of Java maxent package
- A study on the effect of pseudo-absences in SDMs (Barbet-Massin et al. 2012)
- A PDF version of this lesson

Back to learn-r main page

Questions? e-mail me at jcoliver@email.arizona.edu.