A very brief introduction to species distribution models in R

Jeff Oliver

28 September, 2022

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

Species distribution modeling in becoming an increasingly important tool to understand how organisms might respond to current and future environmental changes. There is an ever-growing number of approaches and literature for species distribution models (SDMs), and you are encouraged to check out the Additional Resources section for a few of these resources. The vignette for the `dismo` package is especially useful, and Jeremy Yoder's introduction is another great place to start. In this tutorial, we'll use publicly available data to build, evaluate, and visualize a distribution model for the saguaro cactus.

Getting started

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

Necessary software

The packages necessary for species distribution modeling will likely require additional, non-R software to work properly. Which software will depend on the operating system of your computer.

Linux On Debian Linux systems, you will likely need to install the libgdal-dev package. You can do this through the terminal via `sudo apt-get install libgdal-dev`.

Windows On Windows machines, you should probably install Rtools. You can find downloads and instructions at https://cran.r-project.org/bin/windows/Rtools/.

Mac OS To use the raster package on Mac OS, you'll need to install Xcode Command Line Tools package. You can do this through a terminal via `xcode-select --install`.

Workspace organization

First we need to setup our development environment. Open RStudio and create a new project via:

- File > New Project…
- Select 'New Directory'

- For the Project Type select 'New Project'
- For Directory name, call it something like "r-sdm" (without the quotes)
- For the subdirectory, select somewhere you will remember (like "My Documents" or "Desktop")

We need to create two folders: 'data' will store the data we will be analyzing, and 'output' will store the results of our analyses. In the RStudio console:

```
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 https://tinyurl.com/saguaro-obs; 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")
```

Components of the model

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/")
```

You might see a warning that starts

```
Warning in getData(name = "worldclim", var = "bio", res = 2.5, path = "data/")
```

but you do not need to worry about that right now.

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
```

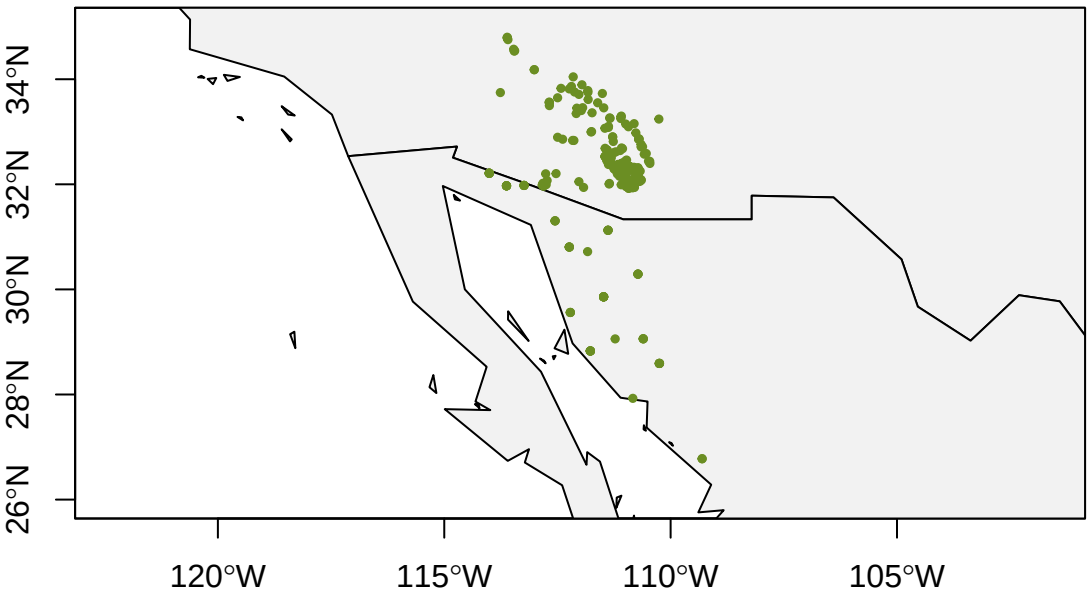
When we look at the `obs_data` data frame now there are no `NA` values, so we are ready to proceed.

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()
```



Looking good!

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[, c("latitude", "longitude")]
```

```
# 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("longitude", "latitude")]

# Build species distribution model
bc_model <- bioclim(x = bioclim_data, p = obs_data)
```

Woo-hoo! No errors here (hopefully).

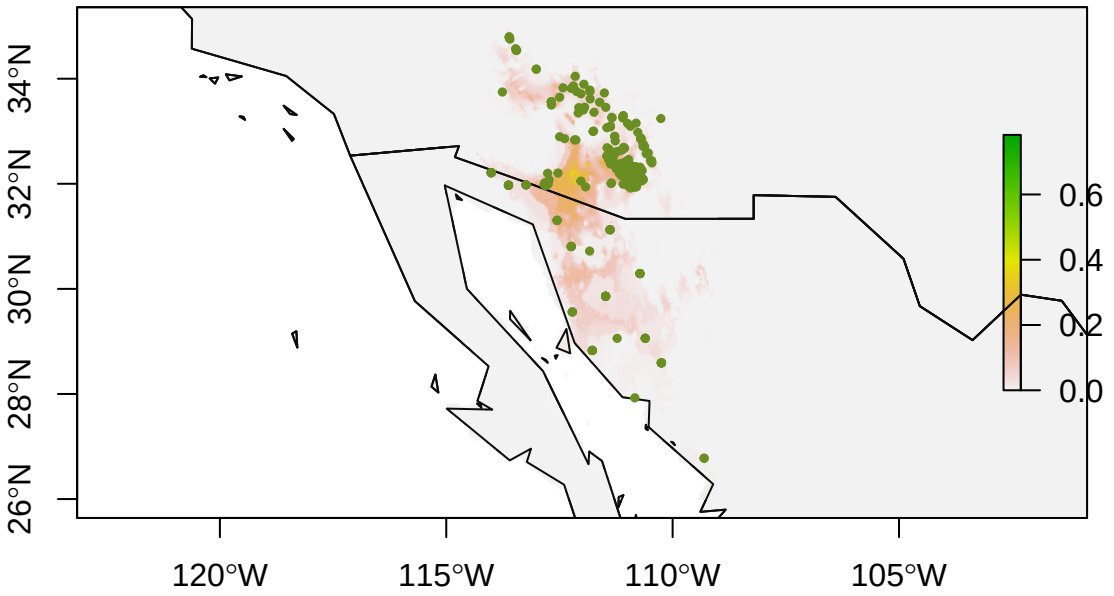
There's one more step we need to take before we plot the model on a map. We need to generate an object that has the model's probability of occurrence for saguaros. We use the `predict` model from the `dismo` package:

```
# Predict presence from model
predict_presence <- dismo::predict(object = bc_model,
                                   x = bioclim_data,
                                   ext = geographic_extent)
```

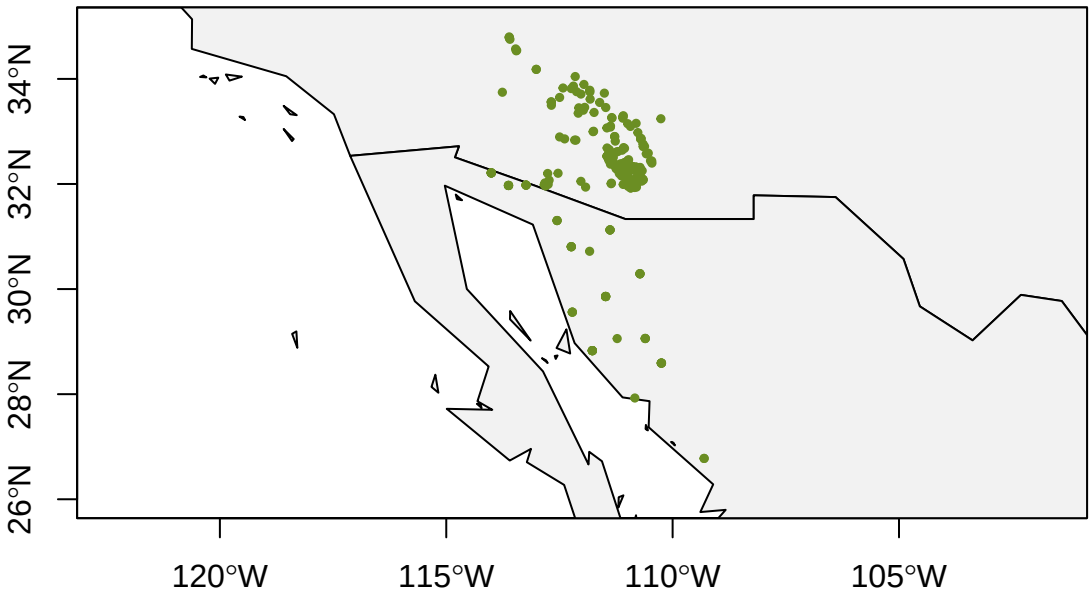
You might be wondering about why we use `dismo::predict` rather than just `predict`. Not surprisingly, different packages sometimes use the same function name to perform very different operations. In the case of `predict`, there are at least three packages loaded into memory that have a `predict` function: dismo, sp, and stats. Although we *probably* would have been fine just using `predict` (R should have used the version from the dismo package), 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 = "grey5")

# Add original observations
points(x = obs_data$longitude,
       y = obs_data$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 Additional Resources 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])

# Set the seed for the random-number generator to ensure results are similar
set.seed(20210707)

# 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:

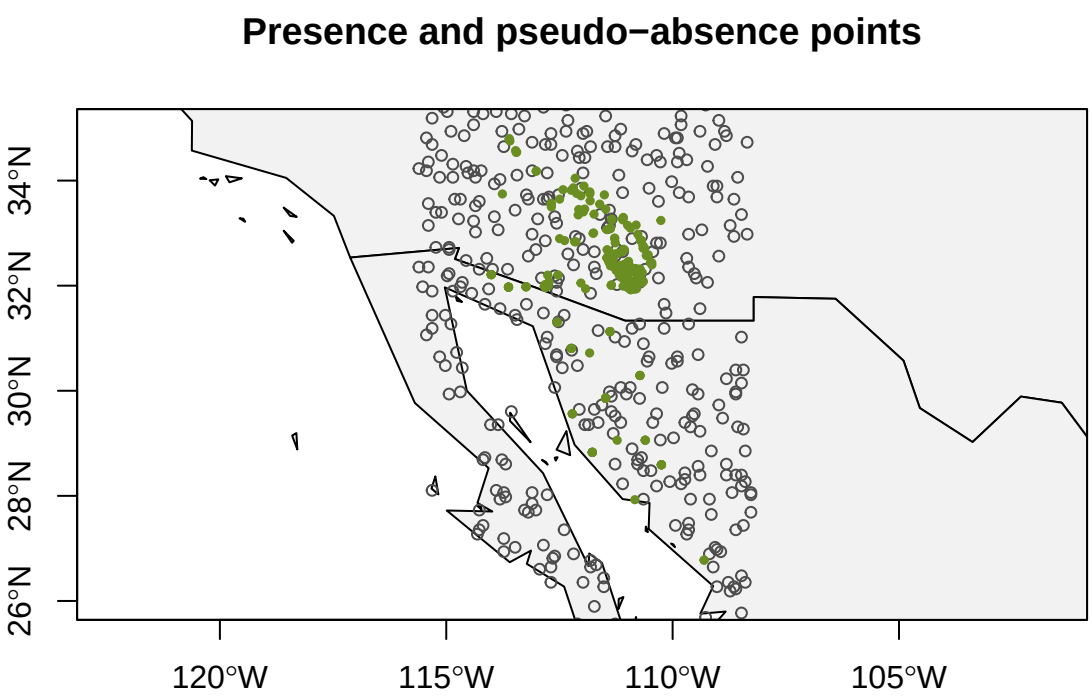
```
head(background)
```

```
##              x        y
```

```
## [1,] -111.1042 33.77083
## [2,] -111.2708 34.85417
## [3,] -108.3958 26.35417
## [4,] -114.9792 35.47917
## [5,] -108.6458 26.27083
## [6,] -111.2708 25.56250
```

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] 2 2 5 1 4 4
```

```
# 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. In this case, we can see that the points have been evenly distributed, with 20% of the points in group 1, our testing group.

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)

# 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)
```

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.

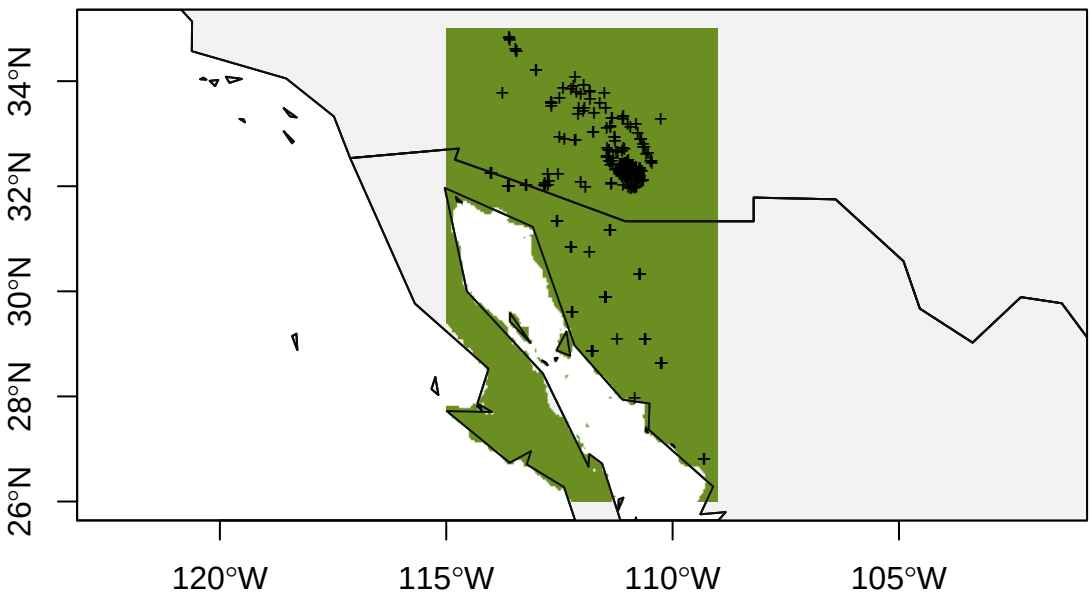
```
# Use testing data for model evaluation
bc_eval <- evaluate(p = presence_test,   # The presence testing data
                    a = background_test, # The absence testing data
                    model = bc_model,    # The model we are evaluating
                    x = bioclim_data)    # Climatic variables for use by model
```

```
# Determine minimum threshold for "presence"
bc_threshold <- threshold(x = bc_eval, stat = "spec_sens")
```

The `threshold` function 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!

```
# 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 = "grey5")
box()
```



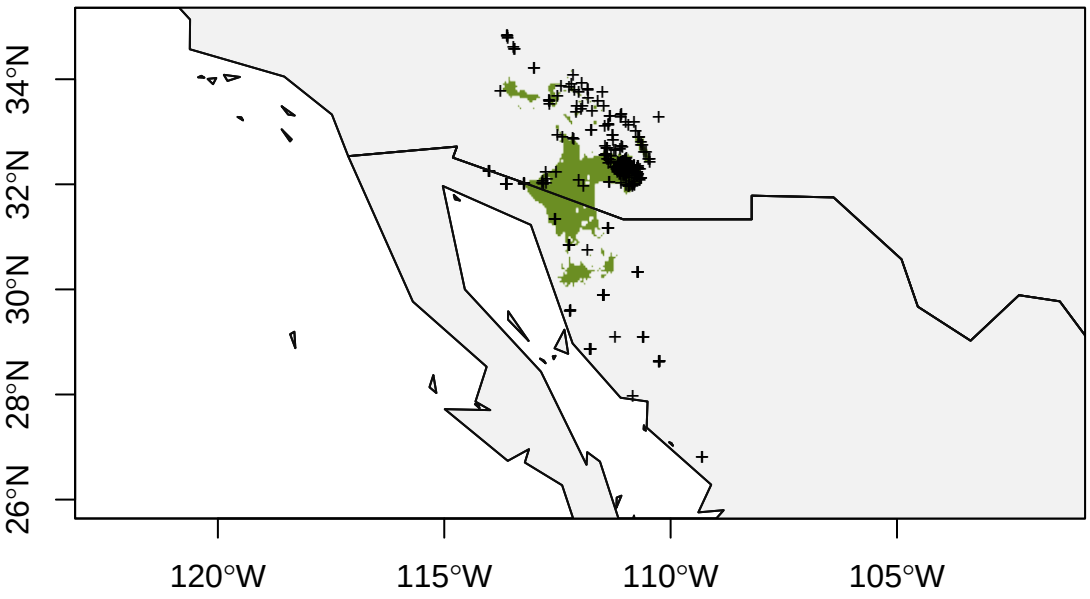
Hmmm…that doesn't look right. It plotted a large portion of the map green. Let's look at what we actually asked R to plot, that is, we plot the value of `predict_presence > bc_threshold`. So what is that?

```
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)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : 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"`):

```
# 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 = "grey5")
box()
```



A final note on our approach: the map we have drawn presents a categorical classification of whether a particular point on the landscape will be suitable or not for the species of interest. This classification relies quite heavily on the value of the threshold (see `bc_threshold` and the documentation for `threshold`) *and* the pseudo-absence points. Given that we used random sampling to generate those pseudo-absence points, there is potential for variation in the predicted range if you run this code more than once (try it! if you re-run the code from the point of creating the pseudo-absence points, you are almost guaranteed a different map.). There are a number of approaches to dealing with this variation, and the paper by Barbet-Massin et al. (2012) is a great resource. I'll leave it as homework for you to determine which approach is most appropriate here!

Our final script, generating the model, determining the threshold, and visualizing the results:

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

rm(list = ls())

# Load additional packages
library("sp")
library("raster")
library("maptools")
library("rgdal")
library("dismo")

# Download bioclim data and store in bioclim_data variable
bioclim_data <- getData(name = "worldclim",
                        var = "bio",
                        res = 2.5,
                        path = "data/")

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

# Drop any rows with NAs
```

```r
obs_data <- obs_data[!is.na(obs_data$latitude), ]

# Only pull out those columns of interest and in the order we want them
obs_data <- obs_data[, c("longitude", "latitude")]

# 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))

# Crop the bioclim data to geographic extent of saguaro
bioclim_data <- crop(x = bioclim_data, y = geographic_extent)

# Create pseudo-absence, or background, points
# 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

# 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

# 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, ]

# Build a model using training data
bc_model <- bioclim(x = bioclim_data, p = presence_train)

# Predict presence from model
predict_presence <- dismo::predict(object = bc_model,
                                   x = bioclim_data,
                                   ext = geographic_extent)
```

```r
# 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")

# Load map data for plotting
data(wrld_simpl)

# 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 = 0.6)

# Redraw those country borders
plot(wrld_simpl, add = TRUE, border = "grey5")
box()
```

Advanced: Forecasting distributions

Now that you have a species distribution model, you can make predictions about the distribution under different climate scenarios. Let us pause for a moment and be very clear about this approach. With all kinds of math wizardry on our side, we are attempting to predict the future. Which means *any* predictions we make should be interpreted with extreme caution. If you are going to go about an approach such as this, it would be wise to run a variety of different models and a variety of different climate scenarios. There are links to such resources in the Additional Resources section, below.

Forecast climate data

We will need to download climate data for the time period of interest. For the purposes of this lesson, we will look a climate projections for the years 2061- 2080. Note there are several different forecast climate models and you can read about the different models at McSweeney et al. 2015 and on the CMIP6 page.

For these data, you will need to manually download forecast climate data from the WorldClim site. You can directly download the data from https://geodata.ucdavis.edu/cmip6/2.5m/MPI-ESM1-2-HR/ssp245/wc2.

1_2.5m_bioc_MPI-ESM1-2-HR_ssp245_2061-2080.tif (the file is a little over 400 MB).

You can do this by using R's `download.file()` function. The file is around 435 MB, so it might take a minute or two to download. By default, `download.file` will timeout after 60 seconds, so if your internet connection is not super-fast, you will want to increase the timeout limit with the `options` function:

```
# Increase timeout to 600 seconds (10 minutes)
options(timeout = max(600, getOption("timeout")))
# Download file and save it to "data" folder
download.file(url = "https://geodata.ucdavis.edu/cmip6/2.5m/MPI-ESM1-2-HR/ssp245/wc2.1_2.5m_bioc_MPI-ESM
              destfile = "data/cmip6-MPI-ESM-HR.tif")
```

This can take a few minutes. Now is a good time to refill your coffee or get a drink of water.

If you see a warning like

> Warning: downloaded length 302055424 != reported length 456981041Warning: URL 'https://geodata.ucdavis.edu/cmip6/2.5m/MPI-ESM1-2-HR/ssp245/wc2.1_2.5m_bioc_MPI-ESM1-2-HR_ssp245_2061-2080.tif': Timeout of 600 seconds was reachedError in download.file(url = "https://geodata.ucdavis.edu/cmip6/2.5m/MPI-ESM1-2-HR/ssp245/wc2.1_2.5m_bioc_MPI-ESM1-2-HR_ssp245_2061-2080.tif", : download from 'https://geodata.ucdavis.edu/cmip6/2.5m/MPI-ESM1-2-HR/ssp245/wc2.1_2.5m_bioc_MPI-ESM1-2-HR_ssp245_2061-2080.tif' failed

Try increasing the timeout value from 600 to 1200 (20 minutes).

We can load the forecast data into memory with the `brick()` function from the raster package:

```
forecast_data <- brick(x = "data/cmip6-MPI-ESM-HR.tif")
```

We need to do one more thing before we can use our data, and that is to make sure the names in our model of bioclimatic variables line up with the names in the forecast data object

```
names(forecast_data) <- names(bioclim_data)
```

Get out the crystal ball

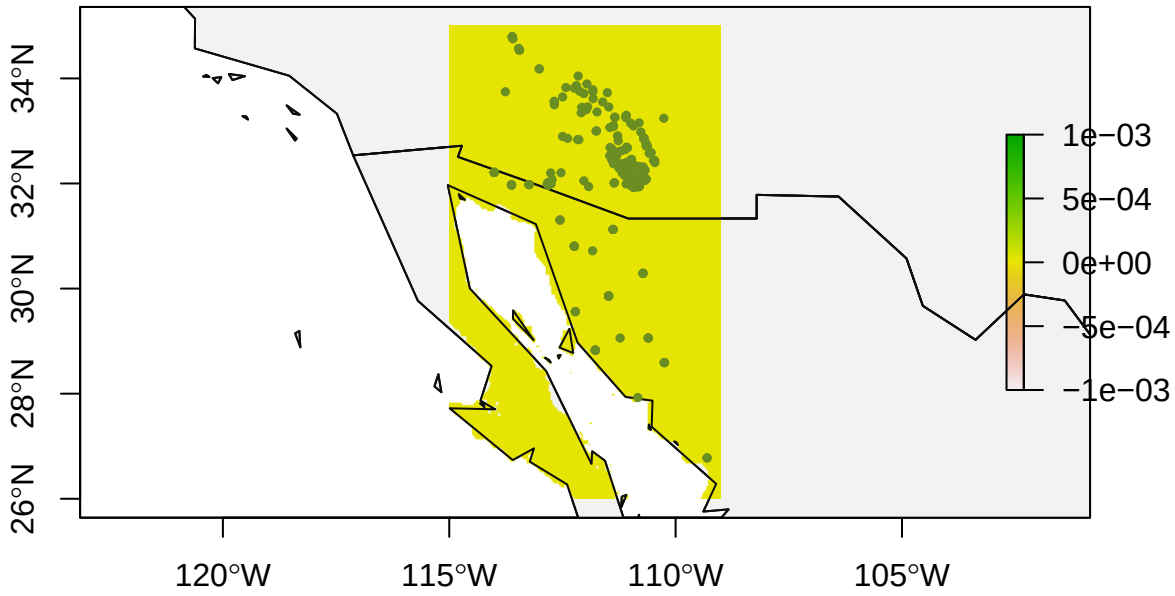
Now that we have the forecast data, we can apply the model we build above, `bc_model`, to the forecast climate data:

```
# Predict presence from model with forecast data
forecast_presence <- dismo::predict(object = bc_model,
                                    x = forecast_data,
                                    ext = geographic_extent)
```

If you want to look at the predicted probabilities of occurrence, you can modify the code we used above.

```
# 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(forecast_presence, add = TRUE)

# Redraw those country borders
plot(wrld_simpl, add = TRUE, border = "grey5")
```

```
# Add original observations
points(x = obs_data$longitude,
       y = obs_data$latitude,
       col = "olivedrab",
       pch = 20,
       cex = 0.75)
box()
```



This doesn't look right. We can look at the distribution of predicted probabilities by typing the name of the variable into the console:

```
forecast_presence
```

```
## 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)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : memory
## names      : layer
## values     : 0, 0  (min, max)
```

Looking and the last line, it says the maximum value is 0. That is *extremely* unlikely. In fact, the reason the probabilities are all zero is because the historical climate data and the forecast climate data are, unfortunately, on different scales. Specifically, the historical climate data for temperatures are all in 10 x degrees Celsius while the forecast climate data for temperatures are in untransformed degrees Celsius. This means that the historical climate data stores a temperature of 20°C as 200, while the forecast climate data stores a temperature of 20°C as 20. Because of this difference, our model was built with data on a drastically different scale than the data we are using to make predictions. There are two solutions:

1. Start at the beginning and transform the *historical* climate data before the model is built. This way, the model is built on data that are on the same scale as the forecast climate data.
2. Transform the *forecast* climate data, so they are on the same scale as the historical climate data on which the model was built.

We will do the second option, transforming the forecast climate data. Note that we only want to transform the temperature climate variables; specifically, this means we need to transform bio1, bio2, and bio4 - bio11.

We use the `message` function to print out the status of the transformation.

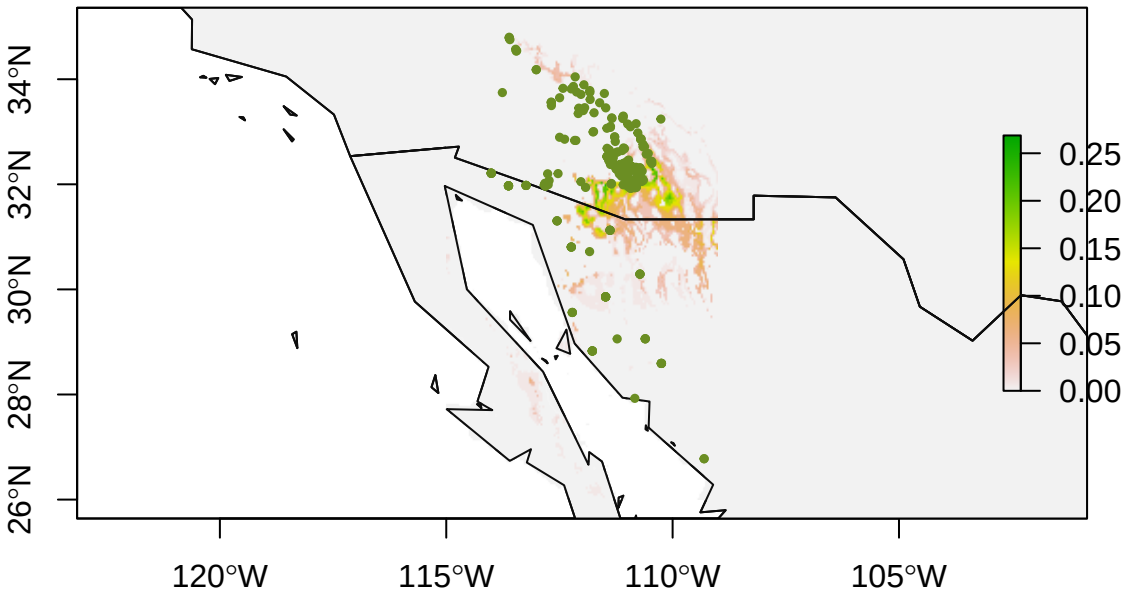
```
# Make a vector of all the variables we need to transform
temp_biovars <- c("bio1", "bio2", "bio4", "bio5", "bio6", "bio7",
                  "bio8", "bio9", "bio10", "bio11")
# Loop over the the variables and multiply each by 10
for (biov in temp_biovars) {
  message("Updating ", biov)
  forecast_data[[biov]] <- 10 * forecast_data[[biov]]
}
```

Now we need to re-run the prediction code with that updated forecast data.

```
# Predict presence from model with forecast data
forecast_presence <- dismo::predict(object = bc_model,
                                    x = forecast_data,
                                    ext = geographic_extent)
```

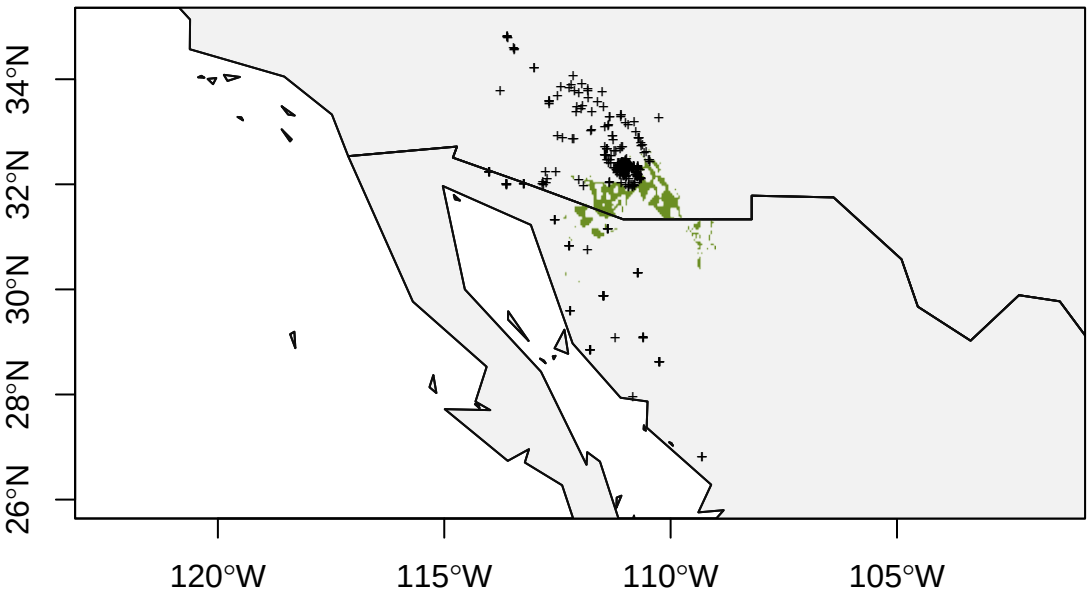
And we plot those predictions as before:

```
# 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(forecast_presence, add = TRUE)

# Redraw those country borders
plot(wrld_simpl, add = TRUE, border = "grey5")

# Add original observations
points(x = obs_data$longitude,
       y = obs_data$latitude,
       col = "olivedrab",
       pch = 20,
       cex = 0.75)
box()
```



We can also map our predictions for presence / absence, using the same threshold that we did for predictions based on current climate data.

```
# 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(forecast_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 = 0.6)

# Redraw those country borders
plot(wrld_simpl, add = TRUE, border = "grey5")
box()
```



Oof. Things do not look great for saguaros under this climate forecast. Try downloading other climate models to see how predictions differ. And remember to interpret these results cautiously.

Additional resources

- The creators of the raster package have an excellent, in-depth guide to species distribution modeling in R
- A lighter-weight introduction to species distribution models in R
- Fast and flexible Bayesian species distribution modelling using Gaussian processes
- 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@arizona.edu.