A Brief Tutorial on Maxent

By Steven J. Phillips, AT&T Research

Last revision: 1/25/2021, to provide additional information regarding permutation importance and percent contribution.

This tutorial gives a basic introduction to use of the MaxEnt program for maximum entropy modelling of species' geographic distributions, written by Steven Phillips, Miro Dudik and Rob Schapire, with support from AT&T Labs-Research, Princeton University, and the Center for Biodiversity and Conservation, American Museum of Natural History. For more details on the theory behind maximum entropy modeling as well as a description of the data used and the main types of statistical analysis used here, see:

Steven J. Phillips, Robert P. Anderson and Robert E. Schapire, **Maximum entropy modeling of species geographic distributions**. *Ecological Modelling*, Vol 190/3-4 pp 231-259, 2006.

Two additional papers describing more recently-added features of the Maxent software are:

Steven J. Phillips and Miroslav Dudik, **Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation**. *Ecography*, Vol 31, pp 161-175, 2008.

Steven J. Phillips et al. Opening the black box: an open-source release of Maxent. Ecography, In press, 2017.

The environmental data we will use consist of climatic and elevational data for South America, together with a potential vegetation layer. Our sample species will be *Bradypus variegatus*, the brown-throated three-toed sloth. These data derive from the 2001 Anderson & Handley taxonomic revision (http://biostor.org/reference/84876) and were used in the Phillips et al. 2006 paper. This tutorial will assume that all the data files are located in the same directory as the maxent program files; otherwise you will need to use the path (e.g., c:\data\maxent\tutorial) in front of the file names used here.

If you would like to reference this tutorial in a publication, report, or online post, an appropriate citation is:

Phillips, S. J. 2017. A Brief Tutorial on Maxent. Available from url: http://biodiversityinformatics.amnh.org/open_source/maxent/. Accessed on XXXX-XX-XX.

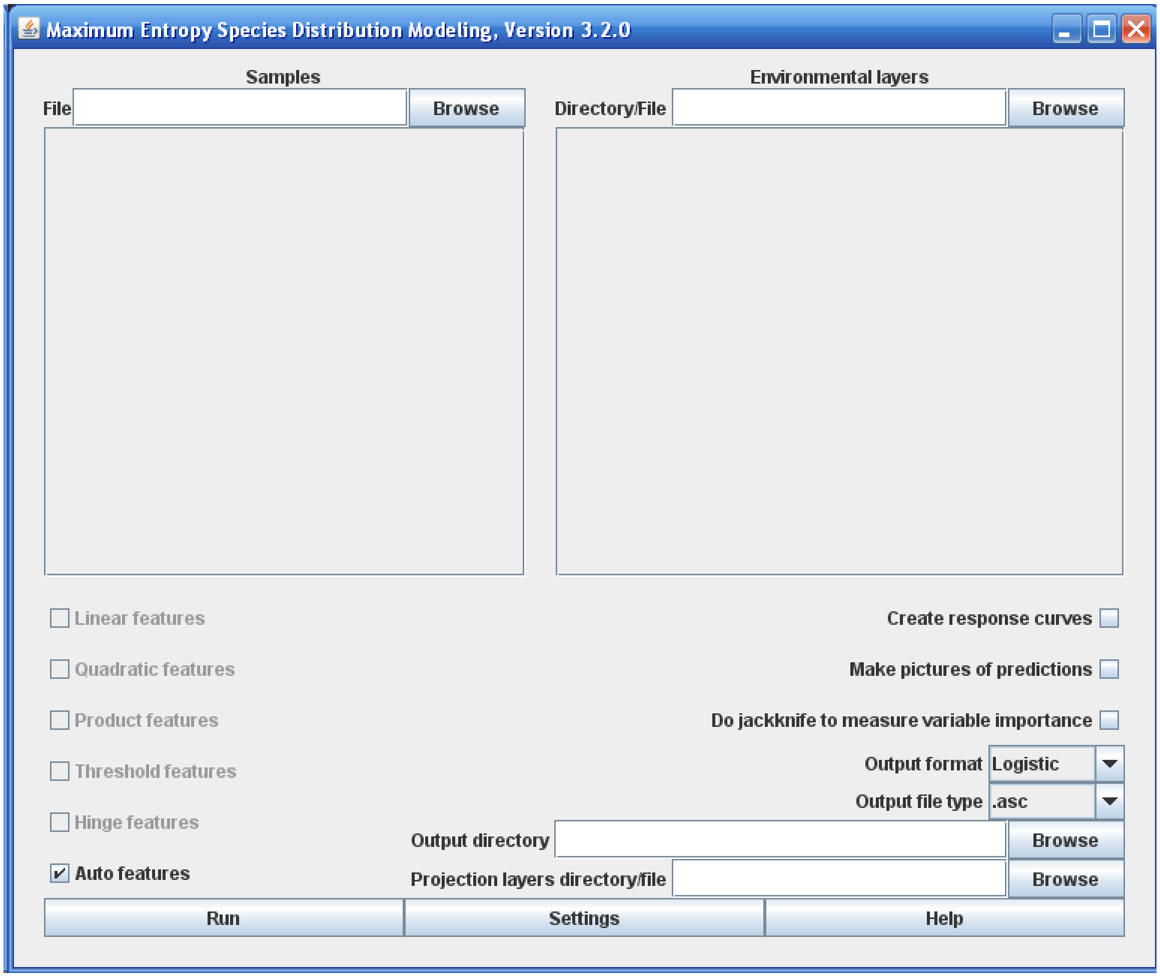
Getting started

Downloading

The software consists of a jar file, maxent.jar, which can be used on any computer running Java version 1.4 or later. Maxent can be downloaded, along with associated literature, from http://biodiversityinformatics.amnh.org/open_source/maxent/; the Java runtime environment can be obtained from java.sun.com/javase/downloads. If you are using Microsoft Windows (as we assume here), you should also download the file maxent.bat, and save it in the same directory as maxent.jar. The website has a file called "readme.txt", which contains instructions for installing the program on your computer.

Firing up

If you are using Microsoft Windows, simply click on the file maxent.bat. Otherwise, enter "java -mx512m -jar maxent.jar" in a command shell (where "512" can be replaced by the megabytes of memory you want made available to the program). The following screen will appear:



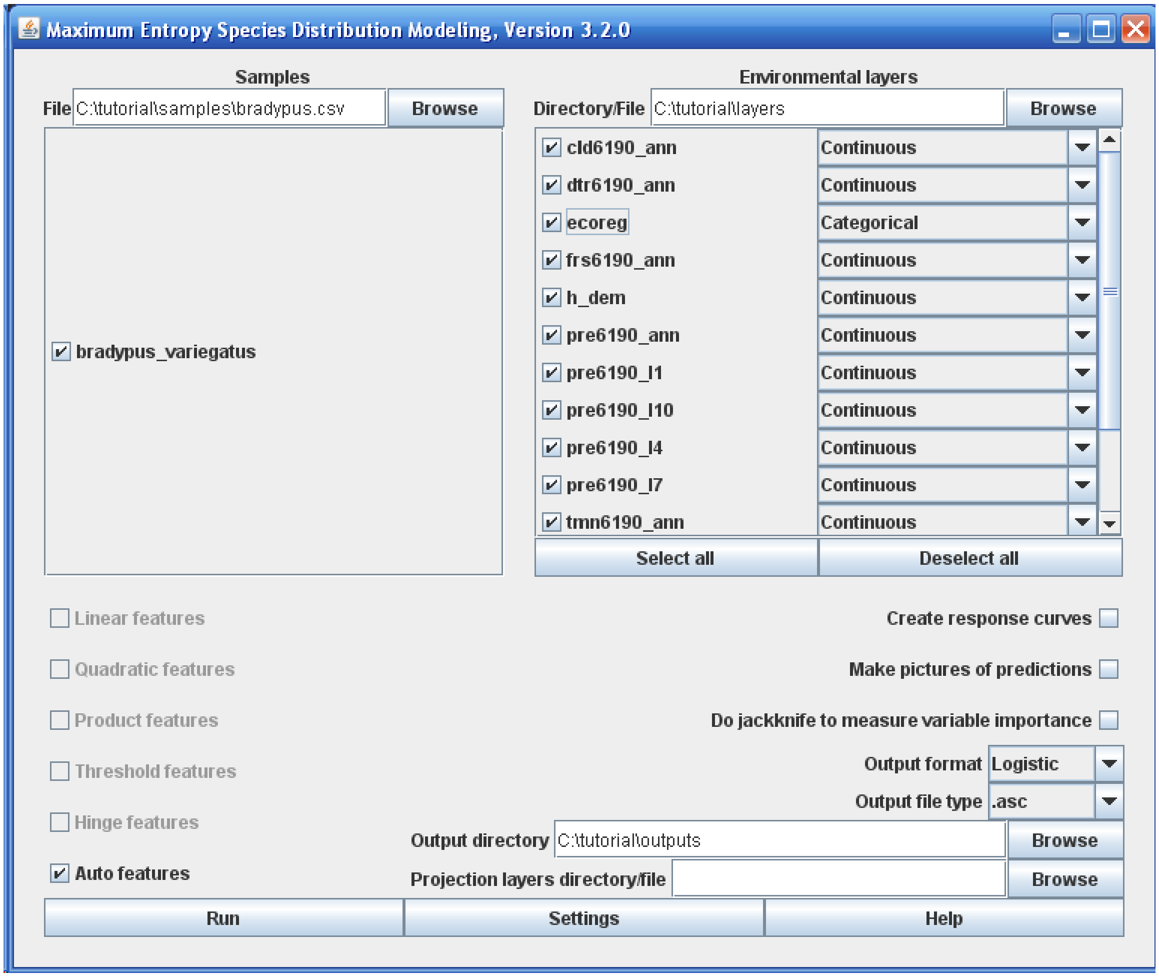
Maximum Entropy Species Distribution Modeling, Version 3.2.0

☐ Linear features
☐ Quadratic features
☐ Product features
☐ Threshold features
☐ Hinge features
☑ Auto features

Create response curves ☐
Make pictures of predictions ☐
Do jackknife to measure variable importance ☐

Output format: Logistic
Output file type: .asc

To perform a run, you need to supply a file containing presence localities ("samples"), a directory containing environmental variables, and an output directory. In our case, the presence localities are in the file "samples\bradypus.csv", the environmental layers are in the directory "layers", and the outputs are going to go in the directory "outputs". You can enter these locations by hand, or browse for them. While browsing for the environmental variables, remember that you are looking for the directory that contains them – you don't need to browse down to the files in the directory. After entering or browsing for the files for *Bradypus*, the program looks like this:



Maximum Entropy Species Distribution Modeling, Version 3.2.0

Samples

File C:\tutorial\samples\bradypus.csv

☑ bradypus_variegatus

Environmental layers

Directory/File C:\tutorial\layers

☑ cld6190_ann	Continuous
☑ dtr6190_ann	Continuous
☑ ecoreg	Categorical
☑ frs6190_ann	Continuous
☑ h_dem	Continuous
☑ pre6190_ann	Continuous
☑ pre6190_l1	Continuous
☑ pre6190_l10	Continuous
☑ pre6190_l4	Continuous
☑ pre6190_l7	Continuous
☑ tmn6190_ann	Continuous

☐ Linear features

☐ Quadratic features

☐ Product features

☐ Threshold features

☐ Hinge features

☑ Auto features

Create response curves ☐

Make pictures of predictions ☐

Do jackknife to measure variable importance ☐

Output format Logistic

Output file type .asc

Output directory C:\tutorial\outputs

Projection layers directory/file

The file "samples\bradypus.csv" contains the presence localities in .csv format. The first few lines are as follows:

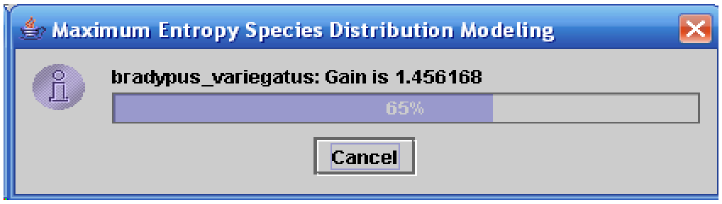
```
species,longitude,latitude
bradypus_variegatus,-65.4,-10.3833
bradypus_variegatus,-65.3833,-10.3833
bradypus_variegatus,-65.1333,-16.8
bradypus_variegatus,-63.6667,-17.45
bradypus_variegatus,-63.85,-17.4
```

There can be multiple species in the same samples file, in which case more species would appear in the panel, along with *Bradypus*. Coordinate systems other than latitude and longitude can be used provided that the samples file and environmental layers use the same coordinate system. The "x" coordinate (longitude, in our case) should come before the "y" coordinate (latitude) in the samples file. If the presence data has duplicate records (multiple records for the same species in the same grid cell), the duplicates are removed by default; this can be changed by clicking on the "Settings" button and deselecting "Remove duplicate presence records".

The directory "layers" contains a number of ascii raster grids (in ESRI's .asc format), each of which describes an environmental variable. **The grids must all have the same geographic bounds and cell size (i.e. all the ascii file headings must match each other perfectly).** One of our variables, "ecoreg", is a categorical variable describing potential vegetation classes. The categories must be indicated by numbers, rather than letters or words. You must tell the program which variables are categorical, as has been done in the picture above.

Doing a run

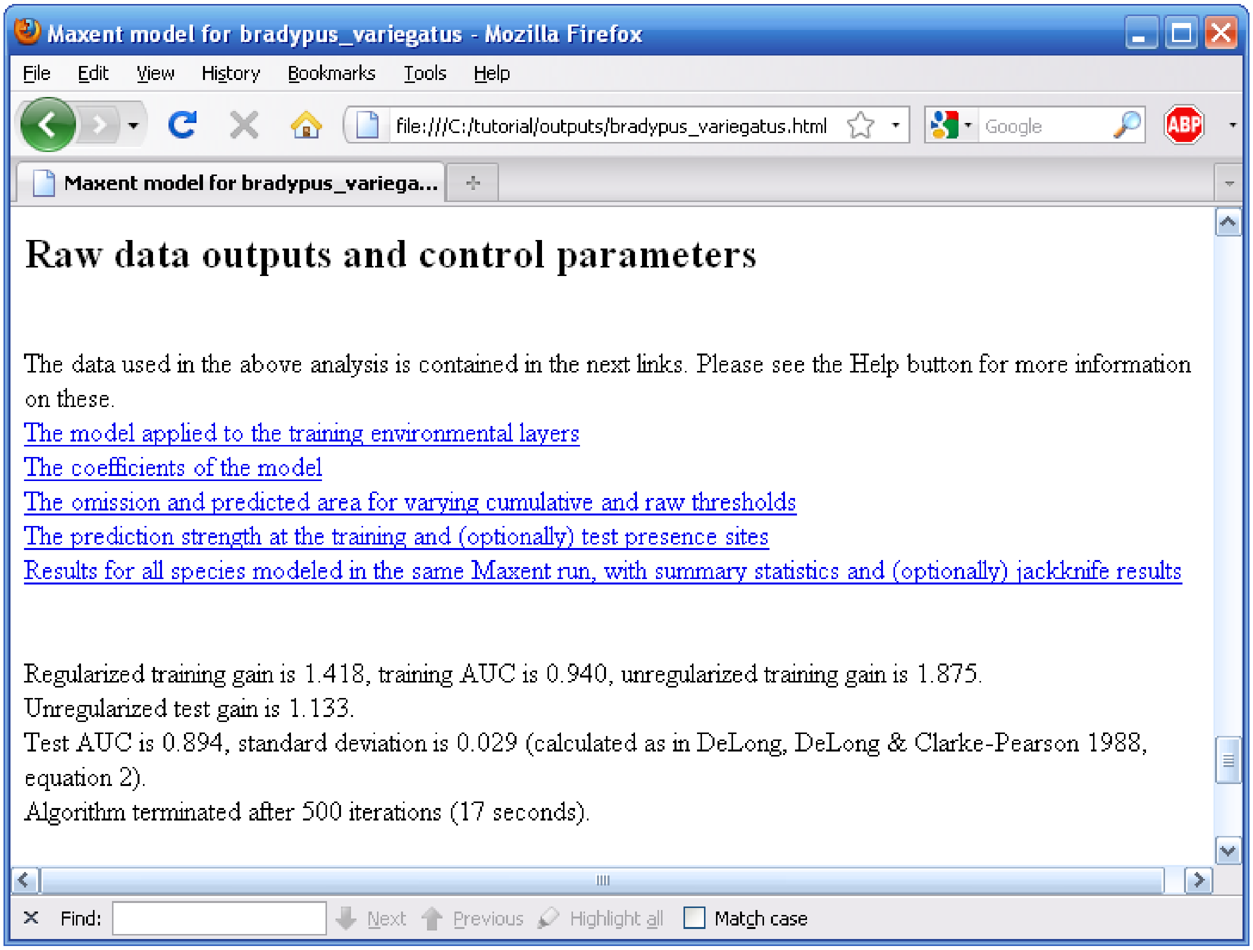
Simply press the “Run” button. A progress monitor describes the steps being taken. After the environmental layers are loaded and some initialization is done, progress towards training of the maxent model is shown like this:


Maximum Entropy Species Distribution Modeling

bradypus_variegatus: Gain is 1.456168

The gain is closely related to deviance, a measure of goodness of fit used in generalized additive and generalized linear models. It starts at 0 and increases towards an asymptote during the run. During this process, Maxent is generating a probability distribution over pixels in the grid, starting from the uniform distribution and repeatedly improving the fit to the data. The gain is defined as the average log probability of the presence samples, minus a constant that makes the uniform distribution have zero gain. At the end of the run, the gain indicates how closely the model is concentrated around the presence samples; for example, if the gain is 2, it means that the average likelihood of the presence samples is exp(2) ≈ 7.4 times higher than that of a random background pixel. Note that Maxent isn’t directly calculating “probability of occurrence”. The probability it assigns to each pixel is typically very small, as the values must sum to 1 over all the pixels in the grid (though we return to this point when we compare output formats).

The run produces multiple output files, of which the most important for analyzing your model is an html file called “bradypus.html”. Part of this file gives pointers to the other outputs, like this:


Maxent model for bradypus_variegatus - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
file:///C:/tutorial/outputs/bradypus_variegatus.html
Google
Maxent model for bradypus_variega...
Find:
Next
Previous
Highlight all
Match case

Raw data outputs and control parameters

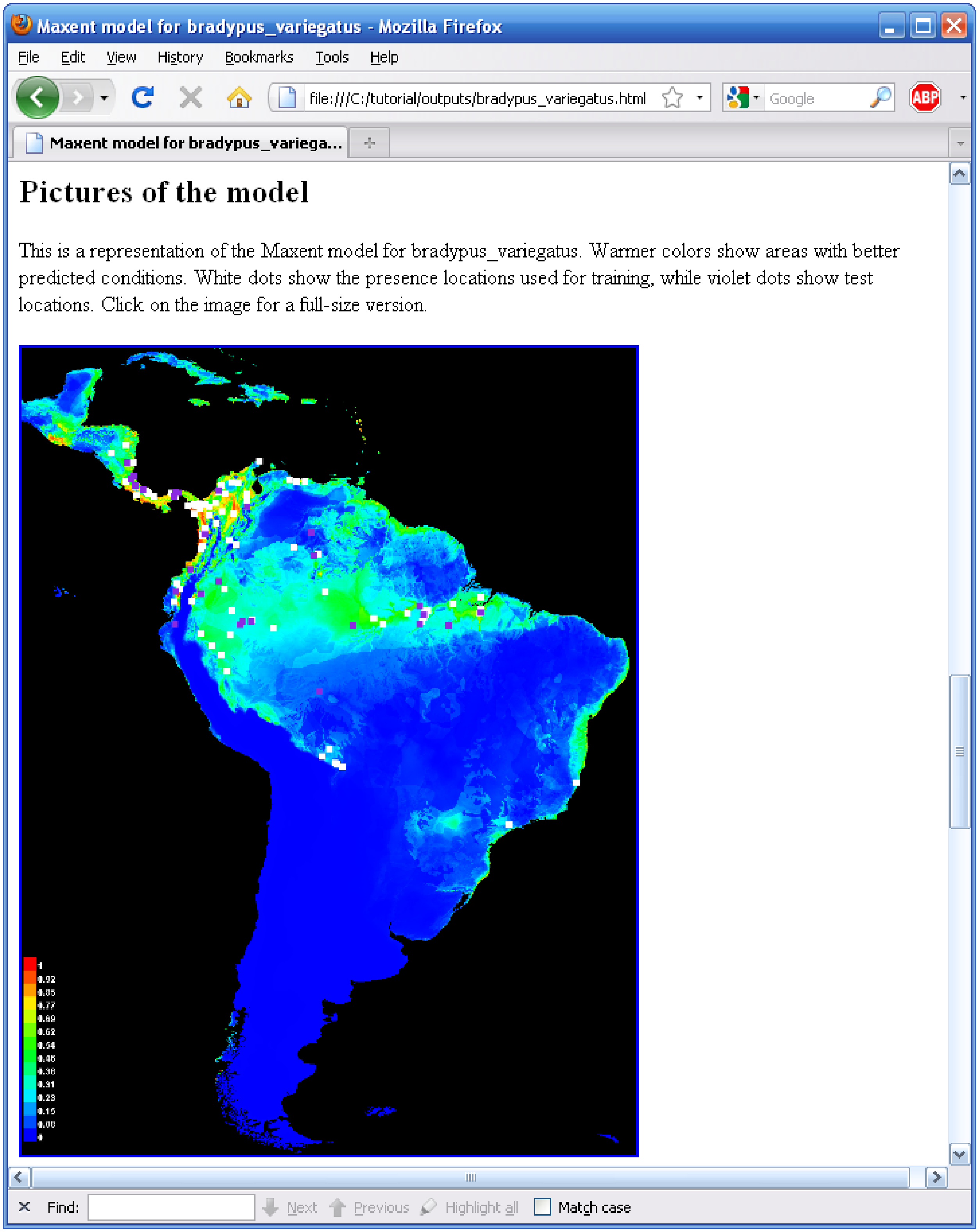
The data used in the above analysis is contained in the next links. Please see the Help button for more information on these.

The model applied to the training environmental layers
The coefficients of the model
The omission and predicted area for varying cumulative and raw thresholds
The prediction strength at the training and (optionally) test presence sites
Results for all species modeled in the same Maxent run, with summary statistics and (optionally) jackknife results

Regularized training gain is 1.418, training AUC is 0.940, unregularized training gain is 1.875.
Unregularized test gain is 1.133.
Test AUC is 0.894, standard deviation is 0.029 (calculated as in DeLong, DeLong & Clarke-Pearson 1988, equation 2).
Algorithm terminated after 500 iterations (17 seconds).

Looking at a prediction

To see what other (more interesting) output there can be in bradpus.html, we will turn on a couple of options and rerun the model. Press the “Make pictures of predictions” button, then click on “Settings”, and type “25” in the “Random test percentage” entry. Then, press the “Run” button again. After the run completes, the file bradypus.html contains a picture like this:


Maxent model for bradypus_variegatus - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
file:///C:/tutorial/outputs/bradypus_variegatus.html
Google
ABP
Maxent model for bradypus_variega...
1
0.92
0.85
0.77
0.69
0.62
0.54
0.46
0.38
0.31
0.23
0.15
0.08
0
Find:
Next
Previous
Highlight all
Match case

Pictures of the model

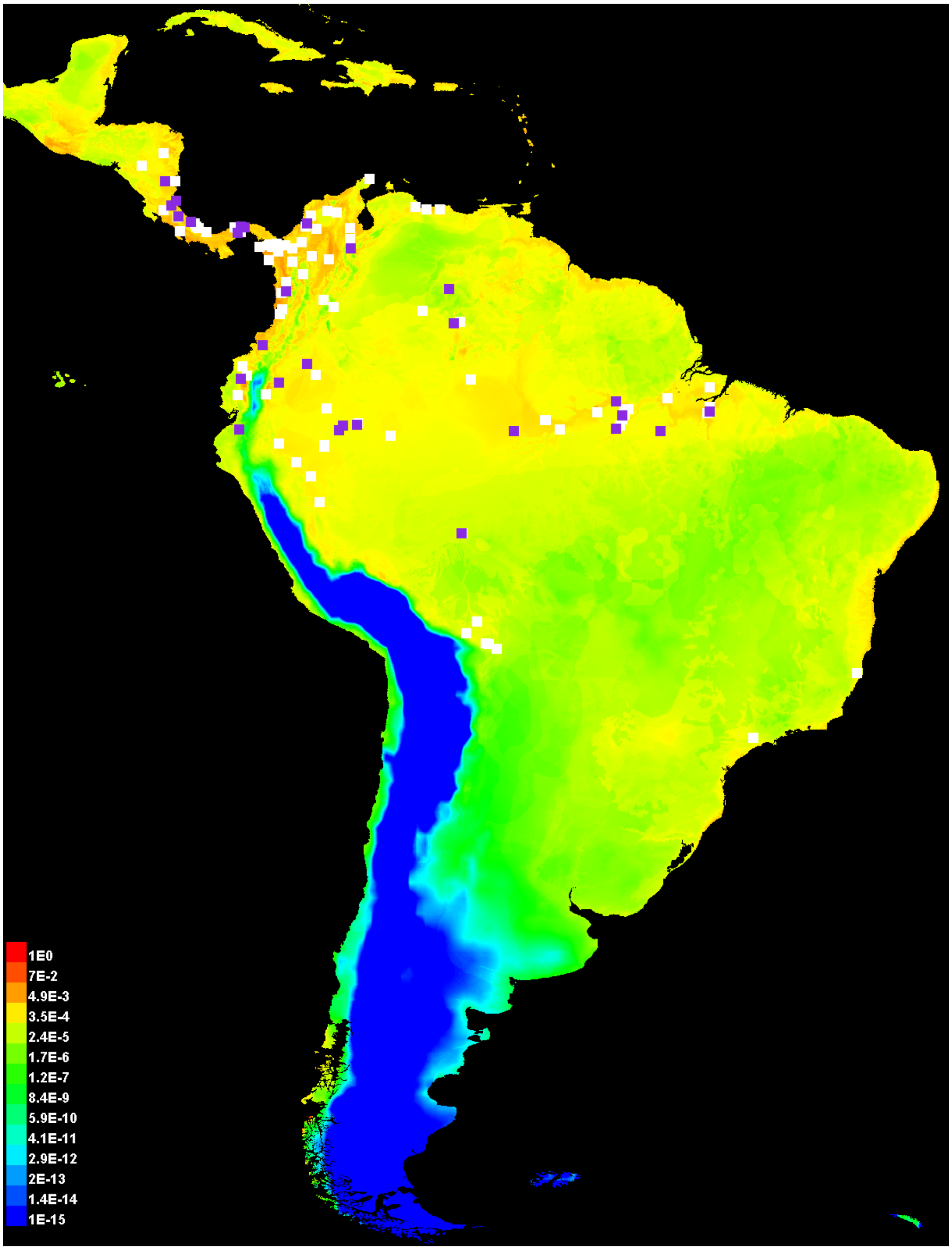
This is a representation of the Maxent model for bradypus_variegatus. Warmer colors show areas with better predicted conditions. White dots show the presence locations used for training, while violet dots show test locations. Click on the image for a full-size version.

The image uses colors to indicate predicted probability that conditions are suitable, with red indicating high probability of suitable conditions for the species, green indicating conditions typical of those where the species is found, and lighter shades of blue indicating low predicted probability of suitable conditions. For *Bradypus*, we see that suitable conditions are predicted to be highly probable through most of lowland Central America, wet lowland areas of northwestern South America, the Amazon basin, Caribean islands, and much of the Atlantic forests in south-eastern Brazil. The file pointed to is an image file (.png) that you can just click on (in Windows) or open in most image processing software. If you want to copy these images, or want to open them with other software, you will find the .png files in the directory called "plots" that has been created as an output during the run.

The test points are a random sample taken from the species presence localities. The same random sample is used each time you run Maxent on the same data set, unless you select the "random seed" option on the settings panel. Alternatively, test data for one or more species can be provided in a separate file, by giving the name of a "Test sample file" in the Settings panel.

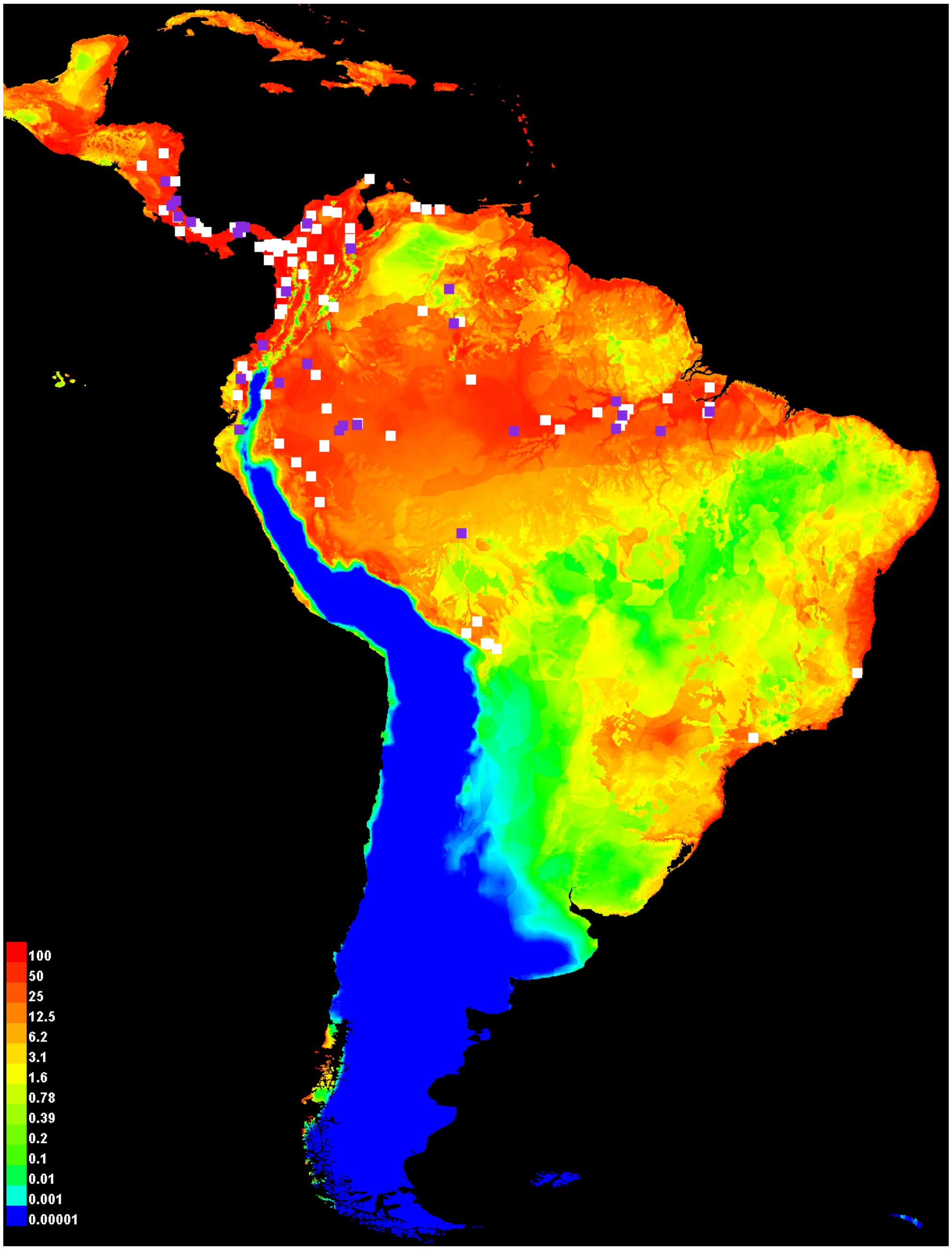
Output formats

Maxent supports four output formats for model values: raw, cumulative, logistic and cloglog. First, the raw output is just the Maxent exponential model itself. Second, the cumulative value corresponding to a raw value of *r* is the percentage of the Maxent distribution with raw value at most *r*. Cumulative output is best interpreted in terms of predicted omission rate: if we set a cumulative threshold of *c*, the resulting binary prediction would have omission rate *c*% on samples drawn from the Maxent distribution itself, and we can predict a similar omission rate for samples drawn from the species distribution. Third, if *c* is the exponential of the entropy of the maxent distribution, then the logistic value corresponding to a raw value of *r* is $c{\cdot}r/(1+c{\cdot}r)$. This is a logistic function, because the raw value is an exponential function of the environmental variables. The cloglog value corresponding to a raw value of r is 1-exp($-c{\cdot}r$). The four output formats are all monotonically related, but they are scaled differently, and have different interpretations. The default output is cloglog, which is the easiest to conceptualize: it gives an estimate between 0 and 1 of probability of presence. Note that probability of presence depends strongly on details of the sampling design, such as the quadrat size and (for vagile organisms) observation time; cloglog output estimates probability of presence assuming that the sampling design is such that typical presence localities have an expected abundance of one individual per quadrat, which results in a probability of presence of about 0.63. The picture of the *Bradypus* model above uses the logistic format, which is very similar to cloglog output, but based on a different theoretical justification. In comparison, using the raw format gives the following picture:



Note that we have used a logarithmic scale for the colors. A linear scale would be mostly blue, with a few red pixels (you can verify this by deselecting "Logscale pictures" on the Settings panel) since the raw format typically gives a small number of sites relatively large values – this can be thought of as an artifact of the raw output being given by an exponential distribution.

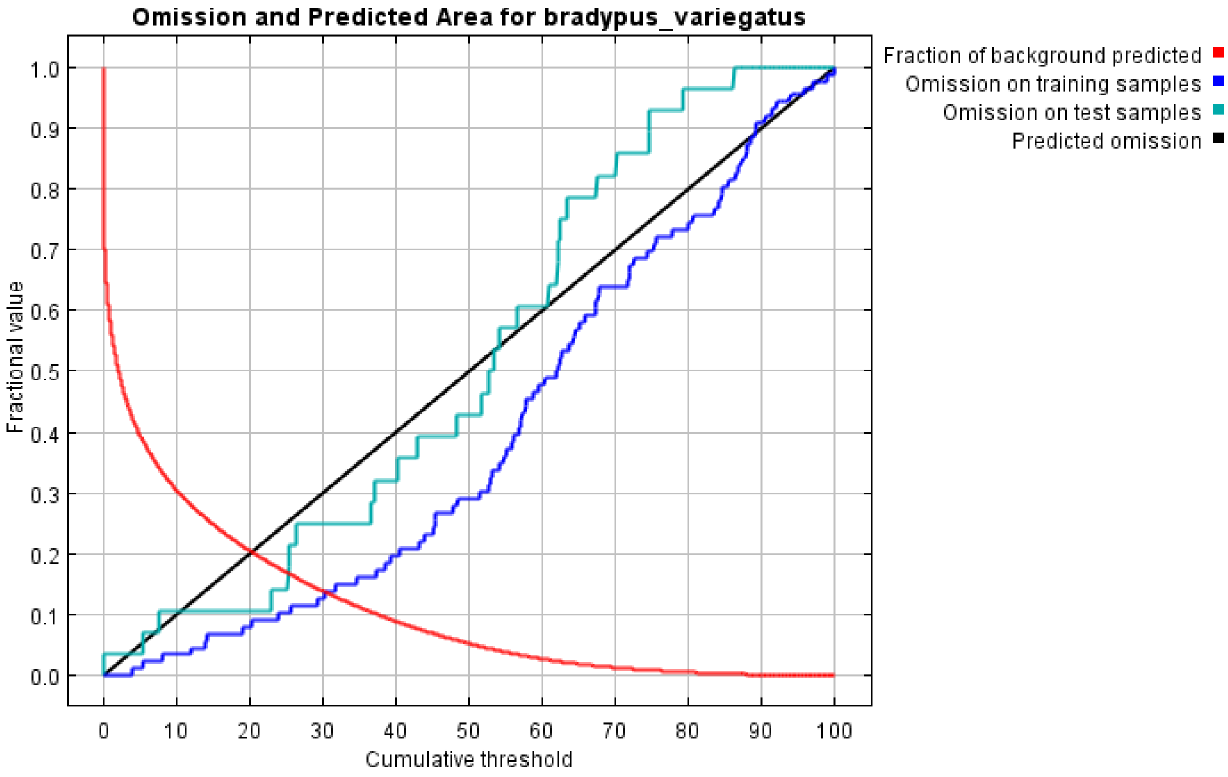
Using the cumulative output format gives the following picture:



As with the raw output, we have used a logarithmic scale for coloring the picture in order to emphasize differences between smaller values. Cumulative output can be interpreted as predicting suitable conditions for the species above a threshold in the approximate range of 1-20 (or yellow through orange, in this picture), depending on the level of predicted omission that is acceptable for the application.

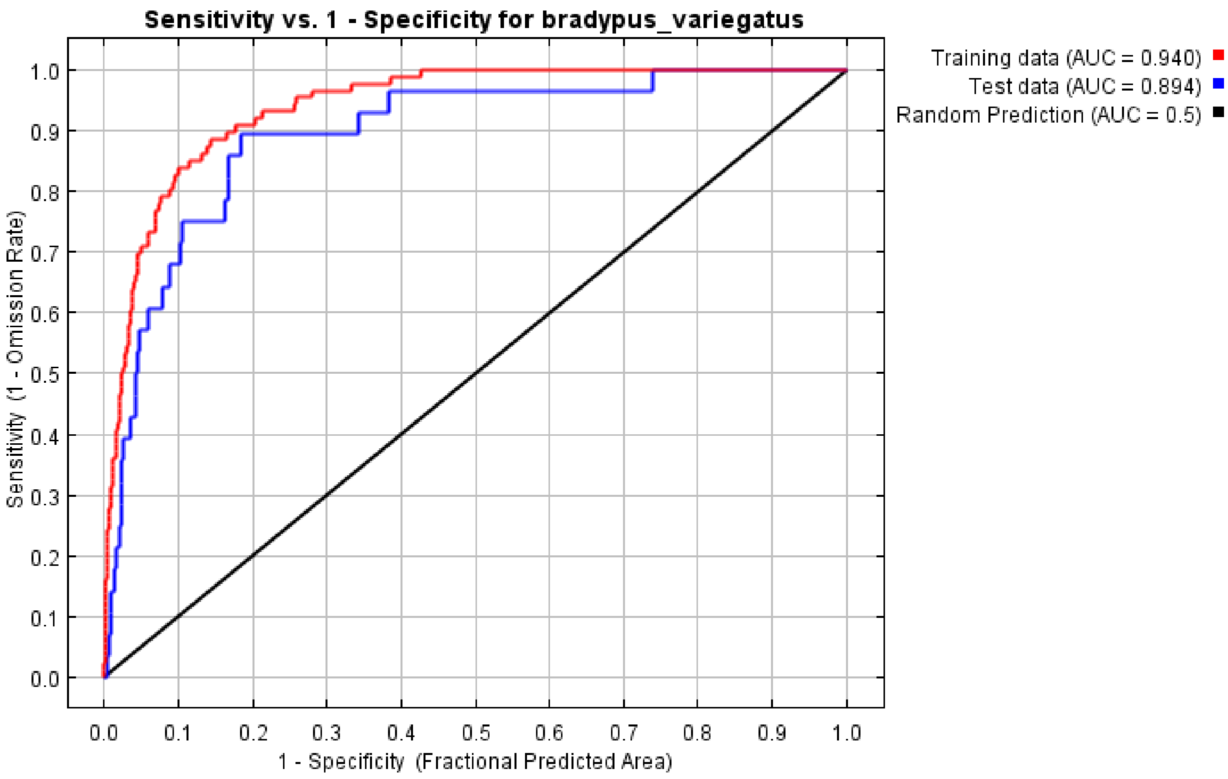
Statistical analysis

The “25” we entered for “random test percentage” told the program to randomly set aside 25% of the sample records for testing. This allows the program to do some simple statistical analysis. Much of the analysis used the use of a threshold to make a binary prediction, with suitable conditions predicted above the threshold and unsuitable below. The first plot shows how testing and training omission and predicted area vary with the choice of cumulative threshold, as in the following graph:



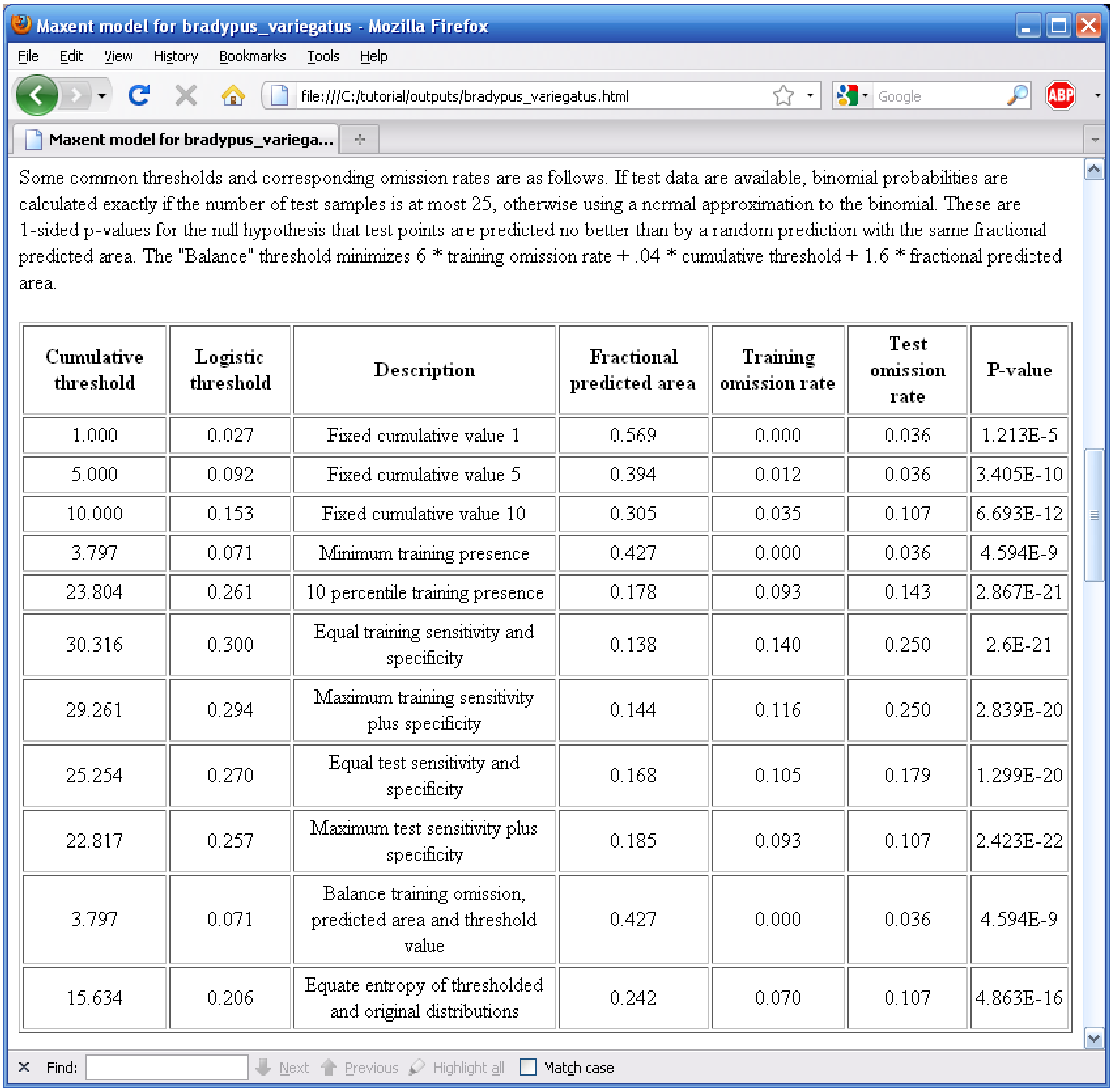
Here we see that the omission on test samples is a very good match to the predicted omission rate, the omission rate for test data drawn from the Maxent distribution itself. The predicted omission rate is a straight line, by definition of the cumulative output format. In some situations, the test omission line lies well below the predicted omission line: a common reason is that the test and training data are not independent, for example if they derive from the same spatially autocorrelated presence data.

The next plot gives the receiver operating curve for both training and test data, shown below. The area under the ROC curve (AUC) is also given here; if test data are available, the standard error of the AUC on the test data is given later on in the web page.



If you use the same data for training and for testing then the red and blue lines will be identical. If you split your data into two partitions, one for training and one for testing it is normal for the red (training) line to show a higher AUC than the blue (testing) line. The red (training) line shows the "fit" of the model to the training data. The blue (testing) line indicates the fit of the model to the testing data, and is the real test of the models predictive power. The turquoise line shows the line that you would expect if your model was no better than random. If the blue line (the test line) falls below the turquoise line then this indicates that your model performs worse than a random model would. The further towards the top left of the graph that the blue line is, the better the model is at predicting the presences contained in the test sample of the data. For more detailed information on the AUC statistic a good starting reference is: Fielding, A.H. & Bell, J.F. (1997) A review of methods for the assessment of prediction errors in conservation presence/ absence models. Environmental Conservation 24(1): 38-49. Because we have only occurrence data and no absence data, "fractional predicted area" (the fraction of the total study area predicted present) is used instead of the more standard commission rate (fraction of absences predicted present). For more discussion of this choice, see the paper in Ecological Modelling mentioned on Page 1 of this tutorial. It is important to note that AUC values tend to be higher for species with narrow ranges, relative to the study area described by the environmental data. This does not necessarily mean that the models are better; instead this behavior is an artifact of the AUC statistic.

If test data are available, the program automatically calculates the statistical significance of the prediction, using a binomial test of omission. For *Bradypus*, this gives:


Maxent model for bradypus_variegatus - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/tutorial/outputs/bradypus_variegatus.html
Google
Maxent model for bradypus_variega...
Find: Next Previous Highlight all Match case

Some common thresholds and corresponding omission rates are as follows. If test data are available, binomial probabilities are calculated exactly if the number of test samples is at most 25, otherwise using a normal approximation to the binomial. These are 1-sided p-values for the null hypothesis that test points are predicted no better than by a random prediction with the same fractional predicted area. The "Balance" threshold minimizes 6 * training omission rate + .04 * cumulative threshold + 1.6 * fractional predicted area.

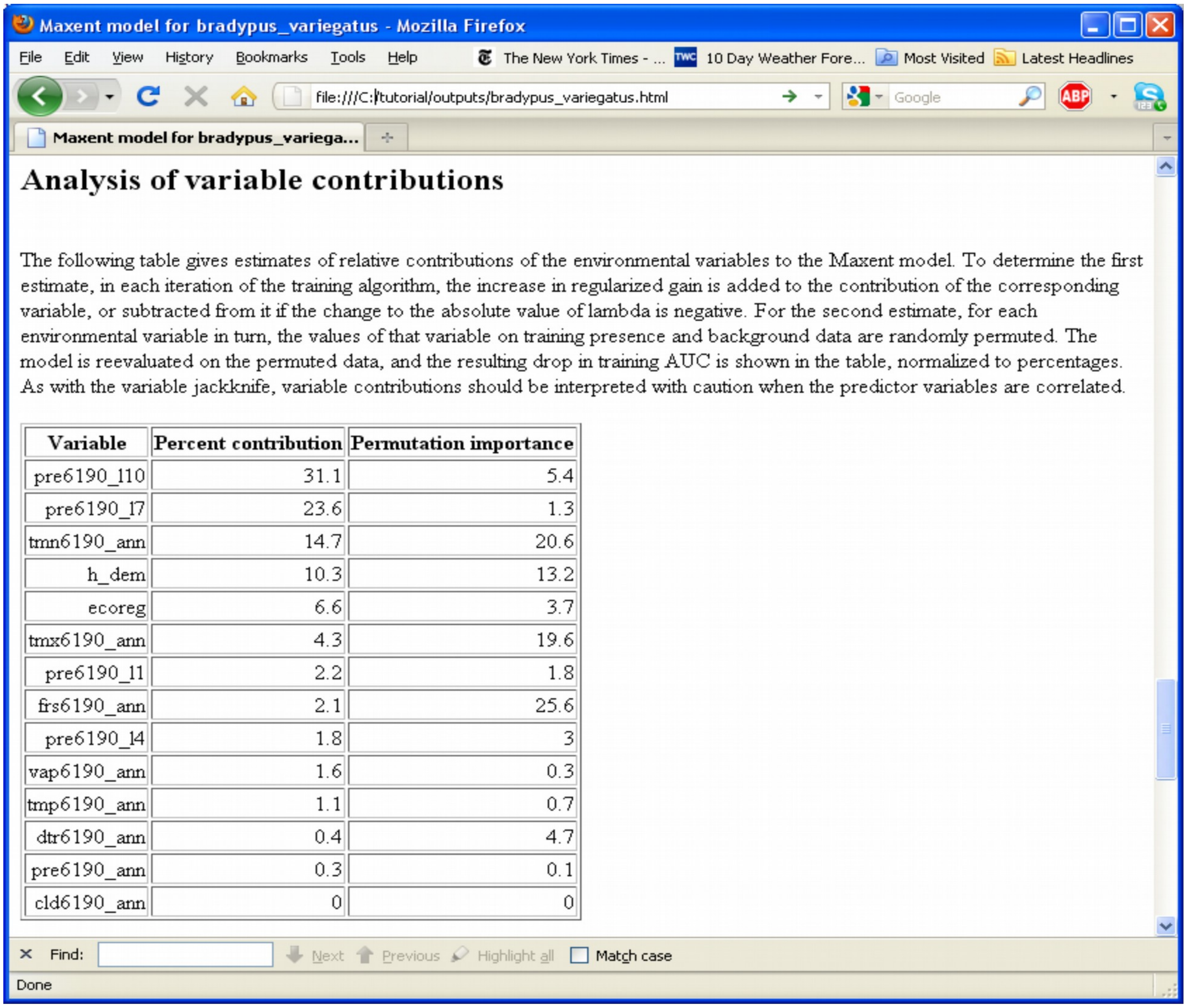
Cumulative threshold	Logistic threshold	Description	Fractional predicted area	Training omission rate	Test omission rate	P-value
1.000	0.027	Fixed cumulative value 1	0.569	0.000	0.036	1.213E-5
5.000	0.092	Fixed cumulative value 5	0.394	0.012	0.036	3.405E-10
10.000	0.153	Fixed cumulative value 10	0.305	0.035	0.107	6.693E-12
3.797	0.071	Minimum training presence	0.427	0.000	0.036	4.594E-9
23.804	0.261	10 percentile training presence	0.178	0.093	0.143	2.867E-21
30.316	0.300	Equal training sensitivity and specificity	0.138	0.140	0.250	2.6E-21
29.261	0.294	Maximum training sensitivity plus specificity	0.144	0.116	0.250	2.839E-20
25.254	0.270	Equal test sensitivity and specificity	0.168	0.105	0.179	1.299E-20
22.817	0.257	Maximum test sensitivity plus specificity	0.185	0.093	0.107	2.423E-22
3.797	0.071	Balance training omission, predicted area and threshold value	0.427	0.000	0.036	4.594E-9
15.634	0.206	Equate entropy of thresholded and original distributions	0.242	0.070	0.107	4.863E-16

For more detailed information on the binomial statistic, see the Ecological Modelling paper mentioned above.

Which variables matter most?

A natural application of species distribution modeling is to answer the question, which variables matter most for the species being modeled? There is more than one way to answer this question; here we outline the possible ways in which Maxent can be used to address it.

While the Maxent model is being trained, we can keep track of which environmental variables are contributing to fitting the model. Each step of the Maxent algorithm increases the gain of the model by modifying the coefficient for a single feature; the program assigns the increase in the gain to the environmental variable(s) that the feature depends on. Converting to percentages at the end of the training process, we get the following table:


Maxent model for bradypus_variegatus - Mozilla Firefox
File Edit View History Bookmarks Tools Help
The New York Times - ...
10 Day Weather Fore...
Most Visited
Latest Headlines
file:///C:/tutorial/outputs/bradypus_variegatus.html
Google
Maxent model for bradypus_variega...
Find:
Next
Previous
Highlight all
Match case
Done

Analysis of variable contributions

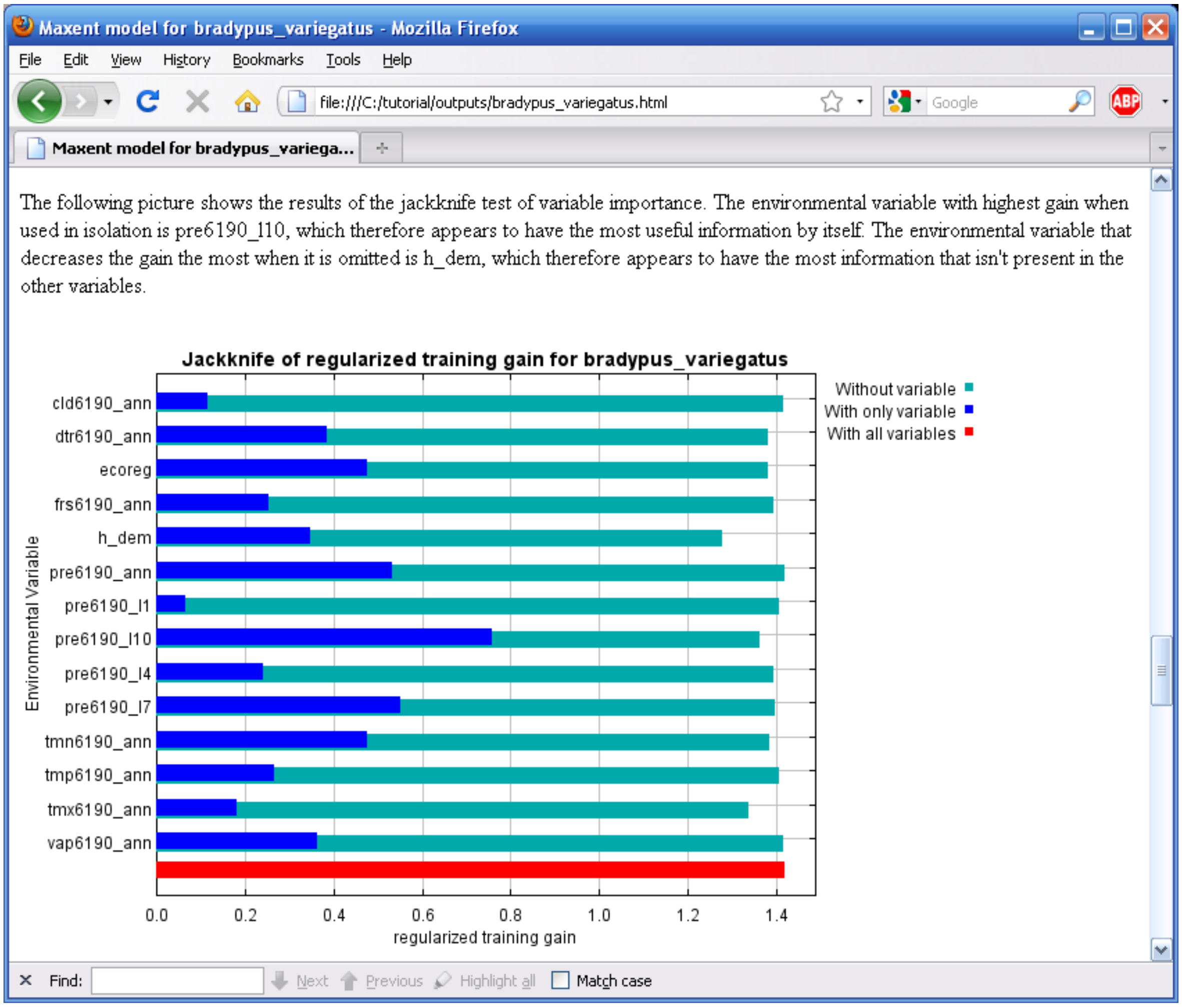
The following table gives estimates of relative contributions of the environmental variables to the Maxent model. To determine the first estimate, in each iteration of the training algorithm, the increase in regularized gain is added to the contribution of the corresponding variable, or subtracted from it if the change to the absolute value of lambda is negative. For the second estimate, for each environmental variable in turn, the values of that variable on training presence and background data are randomly permuted. The model is reevaluated on the permuted data, and the resulting drop in training AUC is shown in the table, normalized to percentages. As with the variable jackknife, variable contributions should be interpreted with caution when the predictor variables are correlated.

Variable	Percent contribution	Permutation importance
pre6190_l10	31.1	5.4
pre6190_l7	23.6	1.3
tmn6190_ann	14.7	20.6
h_dem	10.3	13.2
ecoreg	6.6	3.7
tmx6190_ann	4.3	19.6
pre6190_l1	2.2	1.8
frs6190_ann	2.1	25.6
pre6190_l4	1.8	3
vap6190_ann	1.6	0.3
tmp6190_ann	1.1	0.7
dtr6190_ann	0.4	4.7
pre6190_ann	0.3	0.1
cld6190_ann	0	0

These percent contribution values are only heuristically defined: they depend on the particular path that the Maxent code uses to get to the optimal solution, and a different algorithm could get to the same solution via a different path, resulting in different percent contribution values. In addition, when there are highly correlated environmental variables, the percent contributions should be interpreted with caution. In our *Bradypus* example, annual precipitation is highly correlated with October and July precipitation. Although the above table shows that Maxent used the October precipitation variable more than any other, and hardly used annual precipitation at all, this does not necessarily imply that October precipitation is far more important to the species than annual precipitation.

The right-hand column in the table shows a second measure of variable contributions, called permutation importance. This measure depends only on the final Maxent model, not the path used to obtain it. The contribution for each variable is determined by randomly permuting the values of that variable among the training points (both presence and background) and measuring the resulting decrease in training AUC. A large decrease indicates that the model depends heavily on that variable. Values are normalized to give percentages.

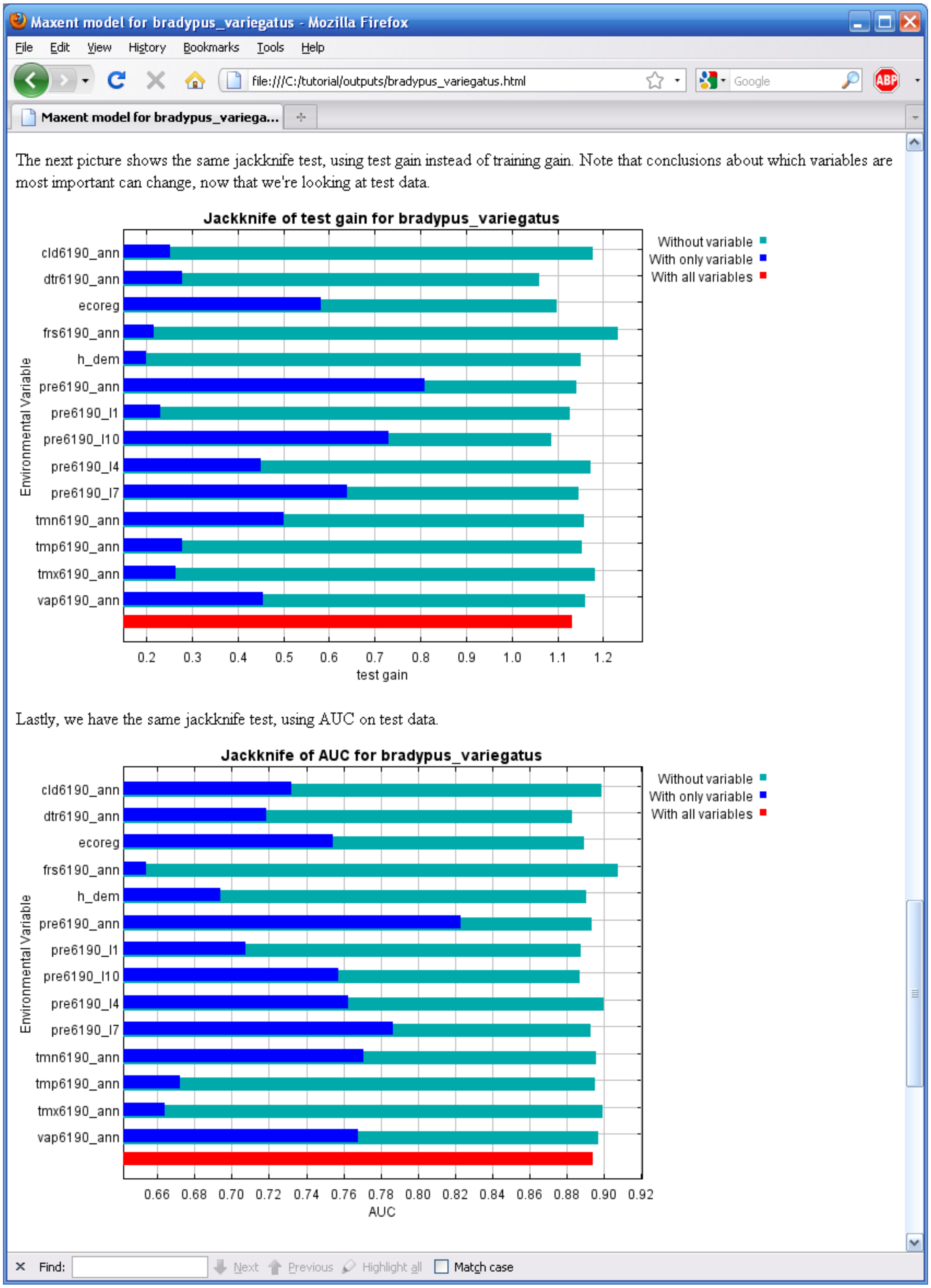
To get alternate estimates of which variables are most important in the model, we can also run a jackknife test by selecting the “Do jackknife to measure variable important” checkbox. When we press the “Run” button again, a number of models are created. Each variable is excluded in turn, and a model created with the remaining variables. Then a model is created using each variable in isolation. In addition, a model is created using all variables, as before. The results of the jackknife appear in the “bradypus.html” files in three bar charts, and the first of these is shown below.



The following picture shows the results of the jackknife test of variable importance. The environmental variable with highest gain when used in isolation is pre6190_l10, which therefore appears to have the most useful information by itself. The environmental variable that decreases the gain the most when it is omitted is h_dem, which therefore appears to have the most information that isn't present in the other variables.

We see that if Maxent uses only pre6190_l1 (average January rainfall) it achieves almost no gain, so that variable is not (by itself) useful for estimating the distribution of *Bradypus*. On the other hand, October rainfall (pre6190_l10) allows a reasonably good fit to the training data. Turning to the lighter blue bars, it appears that no variable contains a substantial amount of useful information that is not already contained in the other variables, because omitting each variable in turn did not decrease the training gain considerably.

The bradypus.html file has two more jackknife plots, which use either test gain or AUC in place of training gain, shown below.



The next picture shows the same jackknife test, using test gain instead of training gain. Note that conclusions about which variables are most important can change, now that we're looking at test data.

Lastly, we have the same jackknife test, using AUC on test data.

Comparing the three jackknife plots can be very informative. The AUC plot shows that annual precipitation (pre6190_ann) is the most effective single variable for predicting the distribution of the occurrence data that was set aside for testing, when predictive performance is measured using AUC, even though it was hardly used by the model built using all variables. The relative importance of annual precipitation also increases in the test gain plot, when compared against the training gain plot. In addition, in the test gain and AUC plots, some of the light blue bars (especially for the monthly precipitation variables) are longer than the red bar, showing that predictive performance improves when the corresponding variables are not used.

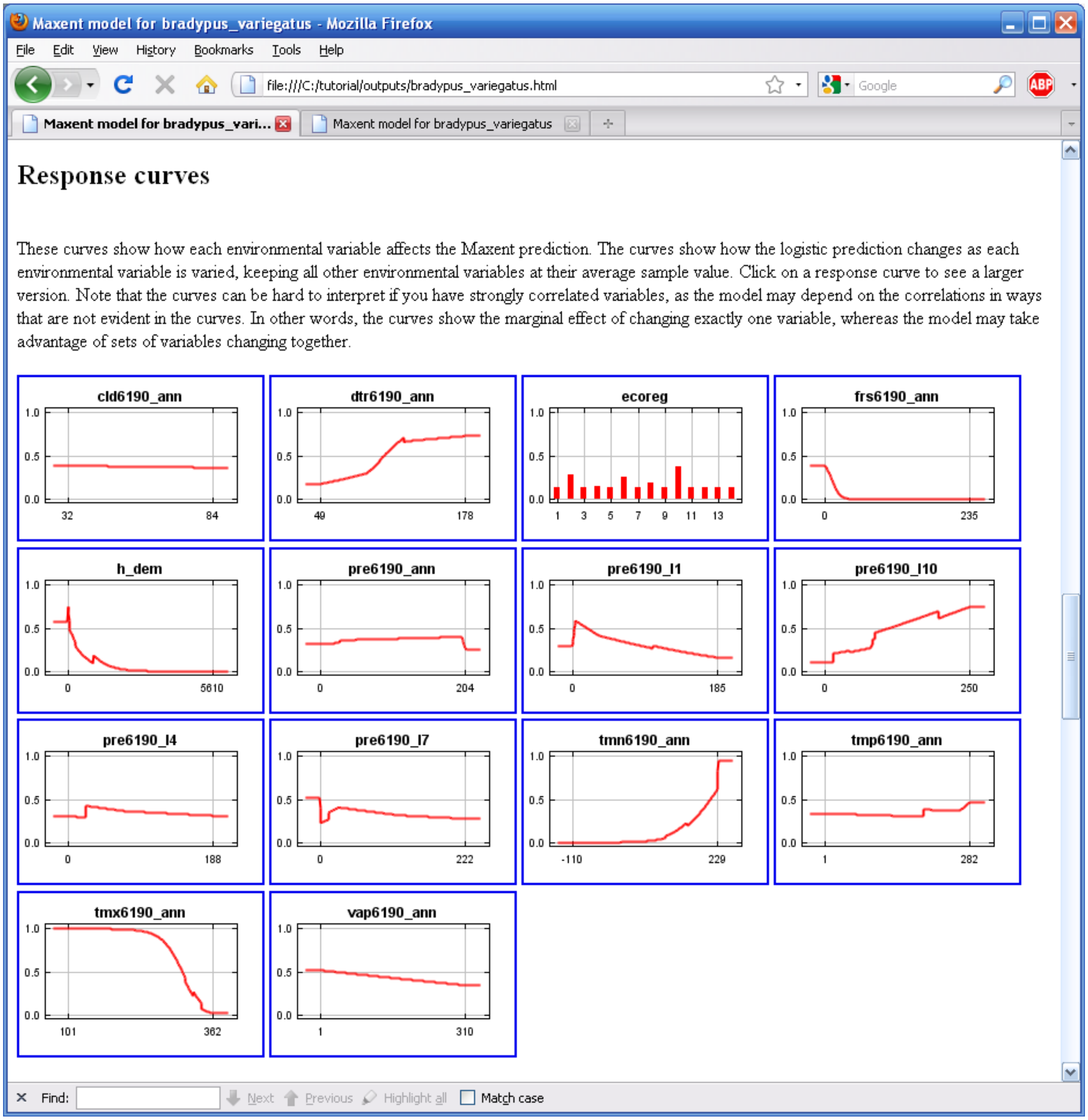
This tells us that monthly precipitation variables are helping Maxent to obtain a good fit to the training data, but the annual precipitation variable generalizes better, giving comparatively better results on the set-aside test data. Phrased differently, models made with the monthly precipitation variables appear to be less transferable. This is important if our goal is to transfer the model, for example by applying the model to future climate variables in order to estimate its future distribution under climate change. It makes sense that monthly precipitation values are less transferable: likely suitable conditions for *Bradypus* will depend not on precise rainfall values in selected months, but on the aggregate average rainfall, and perhaps on rainfall consistency or lack of extended dry periods. When we are modeling on a continental scale, there will probably be shifts in the precise timing of seasonal rainfall patterns, affecting the monthly precipitation but not suitable conditions for *Bradypus*.

In general, it would be better to use variables that are more likely to be directly relevant to the species being modeled. For example, the Worldclim website (www.worldclim.org) provides “BIOCLIM” variables, including derived variables such as “rainfall in the wettest quarter”, rather than monthly values.

A last note on the jackknife outputs: the test gain plot shows that a model made only with January precipitation (pre6190_l1) results in a negative test gain. This means that the model is slightly worse than a null model (i.e., a uniform distribution) for predicting the distribution of occurrences set aside for testing. This can be regarded as more evidence that the monthly precipitation values are not the best choice for predictor variables.

How does the prediction depend on the variables?

Now press the “Create response curves”, deselect the jackknife option, and rerun the model.   This results in the following section being added to the “bradypus.html” file:

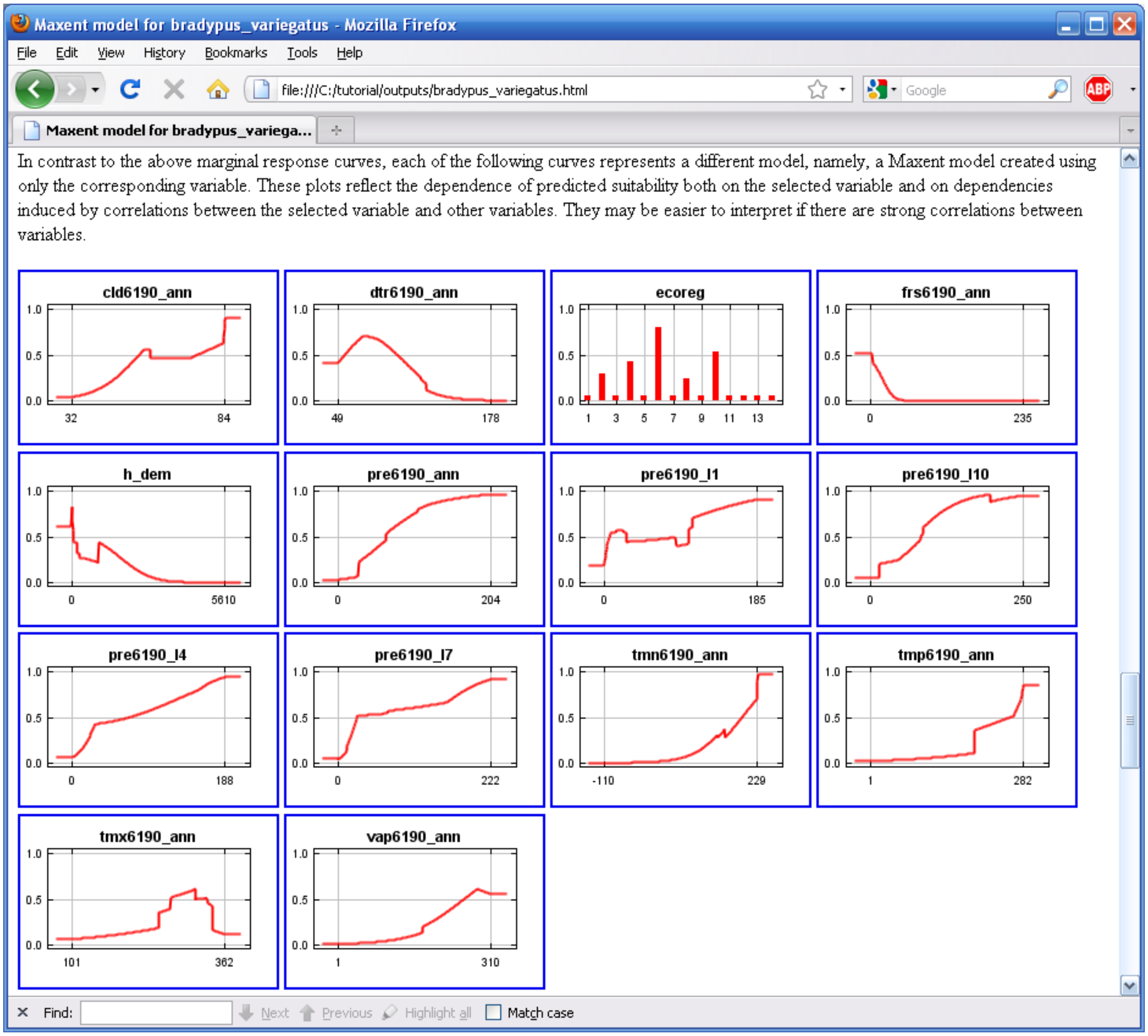


Response curves

These curves show how each environmental variable affects the Maxent prediction. The curves show how the logistic prediction changes as each environmental variable is varied, keeping all other environmental variables at their average sample value. Click on a response curve to see a larger version. Note that the curves can be hard to interpret if you have strongly correlated variables, as the model may depend on the correlations in ways that are not evident in the curves. In other words, the curves show the marginal effect of changing exactly one variable, whereas the model may take advantage of sets of variables changing together.

Each of the thumbnail images can be selected (by clicking on them) to obtain a more detailed plot, and if you would like to copy or open these plots with other software, the .png files can be found in the “plots” directory. Looking at vap6190_ann, we see that the response is low for values of vap6190_ann in the range 1-200, and is higher for values in the range 200-300. The value shown on the y-axis is predicted probability of suitable conditions, as given by the logistic output format, with all other variables set to their average value over the set of presence localities.

Note that if the environmental variables are correlated, as they are here, the marginal response curves can be misleading. For example, if two closely correlated variables have response curves that are near opposites of each other, then for most pixels, the combined effect of the two variables may be small. As another example, we see that predicted suitability is negatively correlated with annual precipitation (pre6190_ann), if all other variables are held fixed. In other words, once the effect of all the other variables has already been accounted for, the marginal effect of increasing annual precipitation is to decrease predicted suitability. However, annual precipitation is highly correlated with the monthly precipitation variables, so in reality we cannot easily hold the monthly values fixed while varying the annual value. The program therefore produces a second set of response curves, in which each curve is made by generating a model using only the corresponding variable, disregarding all other variables:

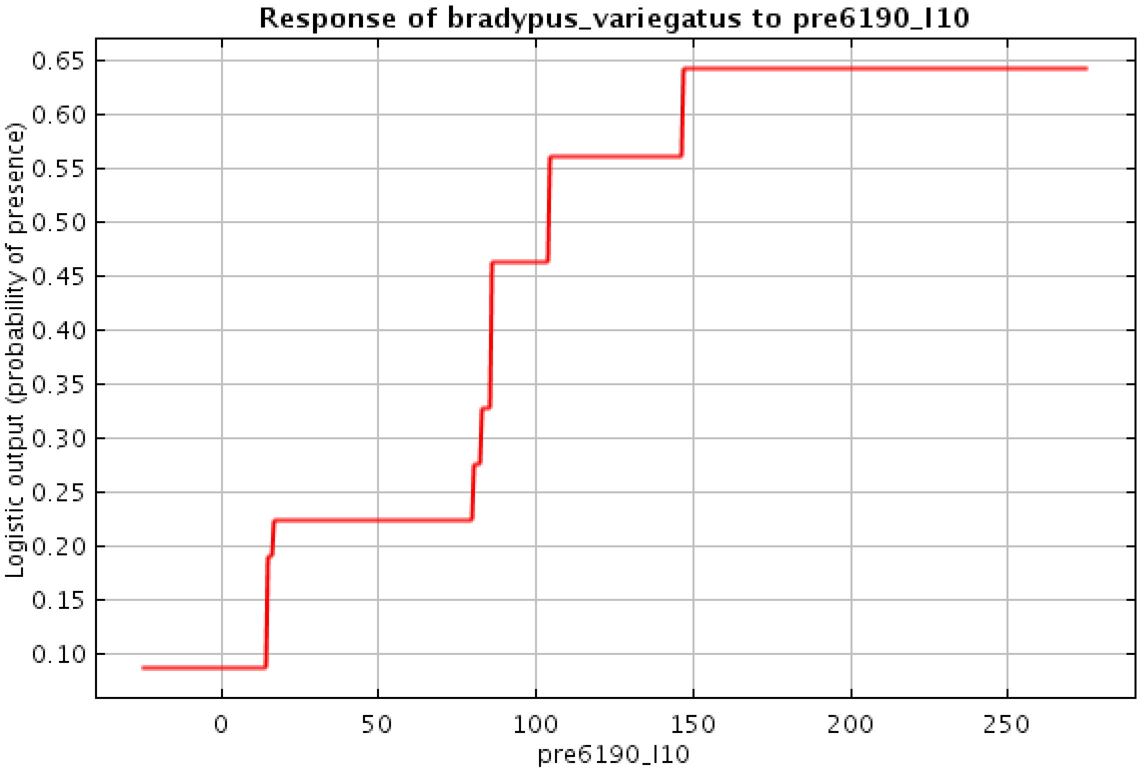


In contrast to the above marginal response curves, each of the following curves represents a different model, namely, a Maxent model created using only the corresponding variable. These plots reflect the dependence of predicted suitability both on the selected variable and on dependencies induced by correlations between the selected variable and other variables. They may be easier to interpret if there are strong correlations between variables.

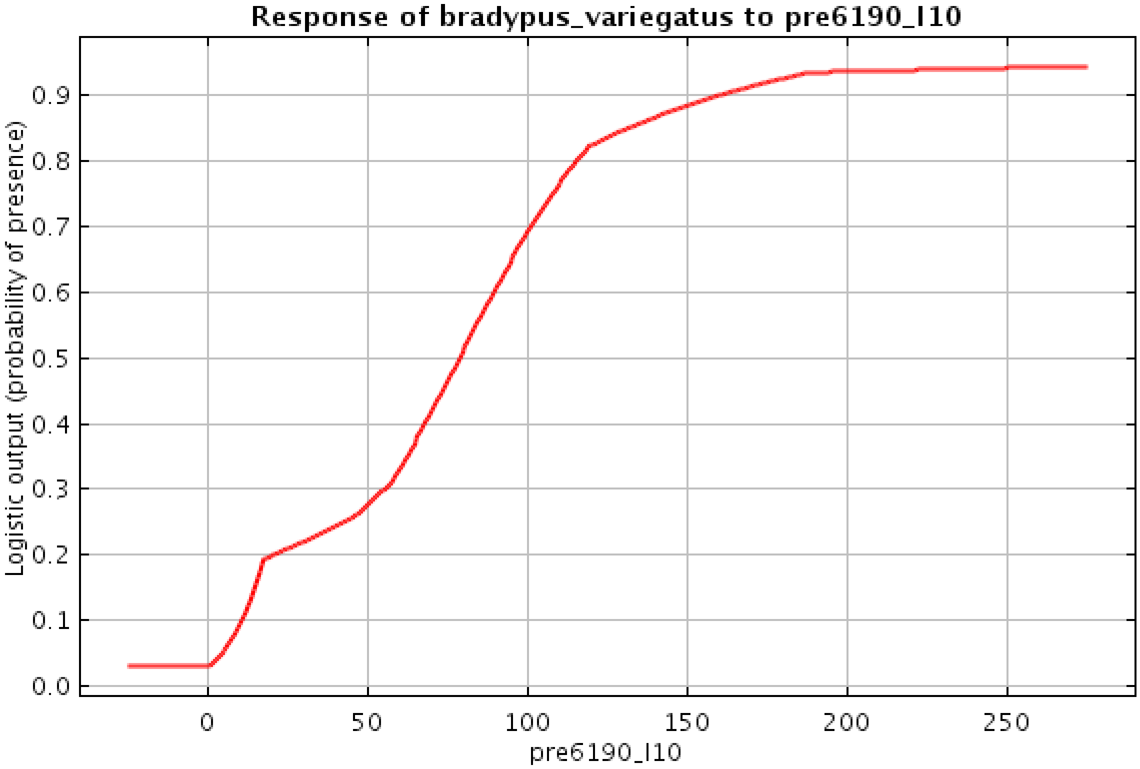
In contrast to the marginal response to annual precipitation in the first set of response curves, we now see that predicted suitability generally increases with increasing annual precipitation.

Feature types and response curves

Response curves allow us to see the difference among different feature types. Deselect the "auto features", select "Threshold features", and press the "Run" button again. Take a look at the resulting feature profiles – you'll notice that they are all step functions, like this one for pre6190_l10:

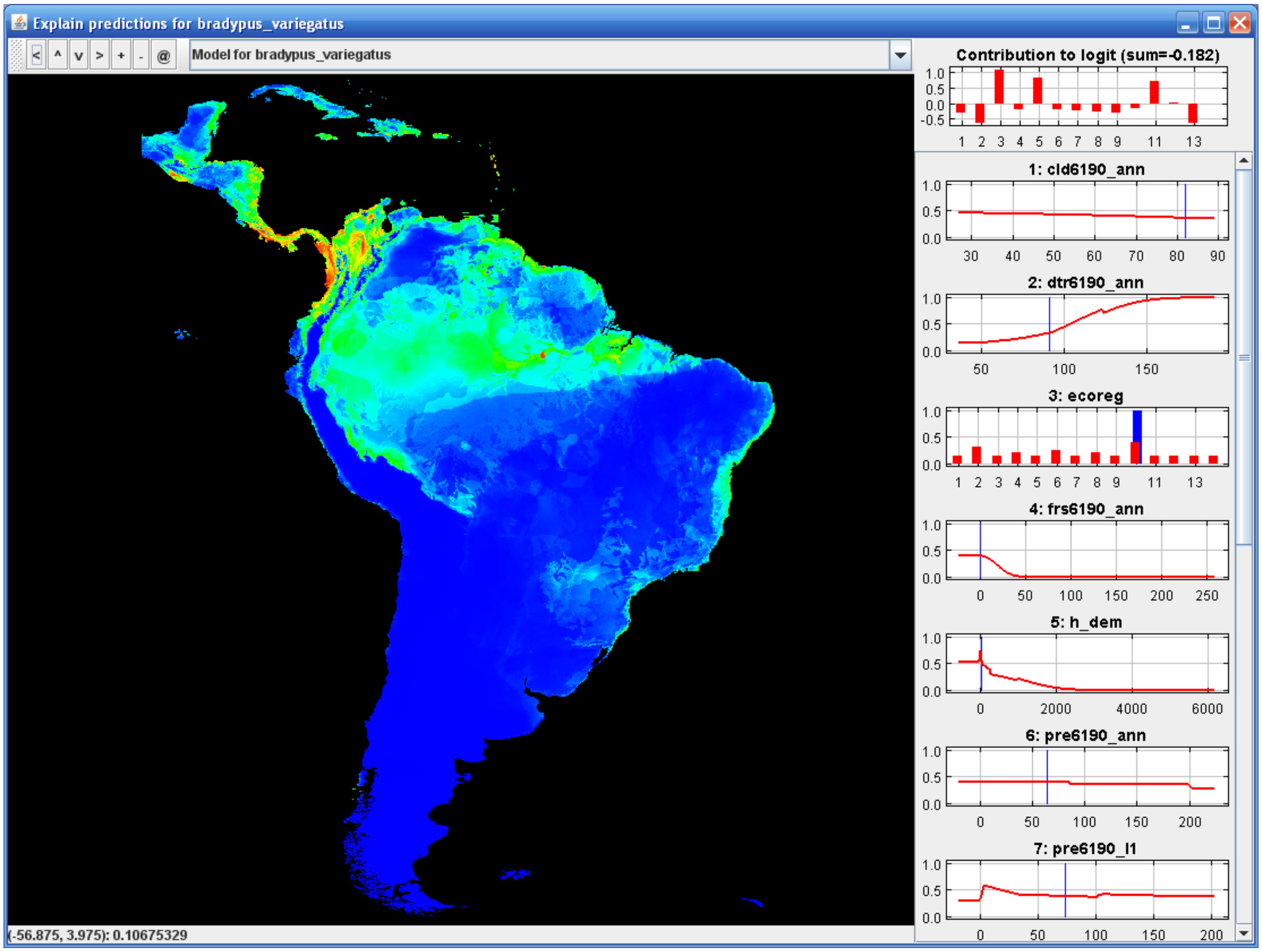


If the same run is done using only hinge features, the resulting feature profile looks like this:



The outlines of the two profiles are similar, but they differ because different feature types allow different possible shapes of response curves. The exponent in a Maxent model is a sum of features, and a sum of threshold features is always a step function, so the logistic output is also a step function (as are the raw and cumulative outputs). In comparison, a sum of hinge features is always a piece-wise linear function, so if only hinge features are used, the Maxent exponent is piece-wise linear. This explains the sequence of connected line segments in the second response curve above. (Note that the lines are slightly curved, especially towards the extreme values of the variable; this is because the logistic output applies a sigmoid function to the Maxent exponent.) Using all classes together (the default, given enough samples) allows many complex responses to be accurately modeled. A deeper explanation of the various feature types can be found by clicking on the help button.

Interactive exploration of predictions



(-56.875, 3.975): 0.10675329

This interactive tool allows you to investigate how Maxent's prediction is determined by the predictor variables across a study area. Clicking on a point on the map shows its location in each response curve. The top right graph shows how much each variable contributes to the logit of the prediction (pointing at a bar on the graph gives the variable name and numerical contribution).

The tool assumes the model is additive (without interactions between variables), so make sure to run it only on the output of a runs without product features. The tool needs data from the response curves created during a Maxent run, so you must select the "Write plot data" option on the Advanced tab for Maxent settings before the run. Then use the following command line from a command window (start → run → cmd) to start the tool:

```
java –cp maxent.jar density.Explain outputs\bradypus_variegatus.asc layers
```

Depending on the directory that you run the command from, you may need to give the full path to the maxent.jar file (e.g., C:\Maxent\maxent.jar), the predictor variables (e.g., C:\maxentTutorial\layers) and the Maxent output grid. Your computer needs enough memory to hold all predictor variables at once.

SWD Format

Another input format can be very useful, especially when your environmental grids are very large. For lack of a better name, it's called "samples with data", or just SWD. The SWD version of our *Bradypus* file, called "bradypus_swd.csv", starts like this:

```
species,longitude,latitude,cld6190_ann,dtr6190_ann,ecoreg,frs6190_ann,h_dem,pre6190_ann,pre6190_l10,pre6190_l1,pre6190_l4,pre6190
_l7,tmn6190_ann,tmp6190_ann,tmx6190_ann,vap6190_ann
bradypus_variegatus,-65.4,-10.3833,76.0,104.0,10.0,2.0,121.0,46.0,41.0,84.0,54.0,3.0,192.0,266.0,337.0,279.0
bradypus_variegatus,-65.3833,-10.3833,76.0,104.0,10.0,2.0,121.0,46.0,40.0,84.0,54.0,3.0,192.0,266.0,337.0,279.0
bradypus_variegatus,-65.1333,-16.8,57.0,114.0,10.0,1.0,211.0,65.0,56.0,129.0,58.0,34.0,140.0,244.0,321.0,221.0
bradypus_variegatus,-63.6667,-17.45,57.0,112.0,10.0,3.0,363.0,36.0,33.0,71.0,27.0,13.0,135.0,229.0,307.0,202.0
bradypus_variegatus,-63.85,-17.4,57.0,113.0,10.0,3.0,303.0,39.0,35.0,77.0,29.0,15.0,134.0,229.0,306.0,202.0
```

It can be used in place of an ordinary samples file. The difference is only that the program doesn't need to look in the environmental layers (the ascii files) to obtain values for the variables at the sample points, instead it reads the values for the environmental variables directly from the table. The environmental layers are thus only used to read the environmental data for the "background" pixels – pixels where the species hasn't necessarily been detected. In fact, the background pixels can also be specified in a SWD format file. The file "background.csv" contains 10,000 background data points. The first few look like this:

```
background,-61.775,6.175,60.0,100.0,10.0,0.0,747.0,55.0,24.0,57.0,45.0,81.0,182.0,239.0,300.0,232.0
background,-66.075,5.325,67.0,116.0,10.0,3.0,1038.0,75.0,16.0,68.0,64.0,145.0,181.0,246.0,331.0,234.0
background,-59.875,-26.325,47.0,129.0,9.0,1.0,73.0,31.0,43.0,32.0,43.0,10.0,97.0,218.0,339.0,189.0
background,-68.375,-15.375,58.0,112.0,10.0,44.0,2039.0,33.0,67.0,31.0,30.0,6.0,101.0,181.0,251.0,133.0
background,-68.525,4.775,72.0,95.0,10.0,0.0,65.0,72.0,16.0,65.0,69.0,133.0,218.0,271.0,346.0,289.0
```

We can run Maxent with "bradypus_swd.csv" as the samples file and "background.csv" (both located in the "swd" directory) as the environmental layers file. Try running it – you'll notice that it runs much faster, because it doesn't have to load the large environmental grids. Another advantage is that you can associate different records with environmental conditions from different time periods. For example, two occurrences recorded 100 years apart from the same grid cell probably reflect considerable variation in environmental conditions, but unless you use SWD format, both records would be given the same environmental variables values. The downside is that it can't make pictures or output grids, because it doesn't have all the environmental data. The way to get around this is to use a "projection", described below.

Batch running

Sometimes you need to generate multiple models, perhaps with slight variations in the modeling parameters or the inputs. Generation of models can be automated with command-line arguments, obviating the need to click and type repetitively at the program interface. The command line arguments can either be given from a command window (a.k.a. shell), or they can be defined in a batch file. Take a look at the file "batchExample.bat" (for example, right click on the .bat file inWindows Explorer and open it using Notepad). It contains the following line:

```
java -mx512m -jar maxent.jar environmentallayers=layers togglelayertype=ecoreg
samplesfile=samples\bradypus.csv outputdirectory=outputs redoifexists autorun
```

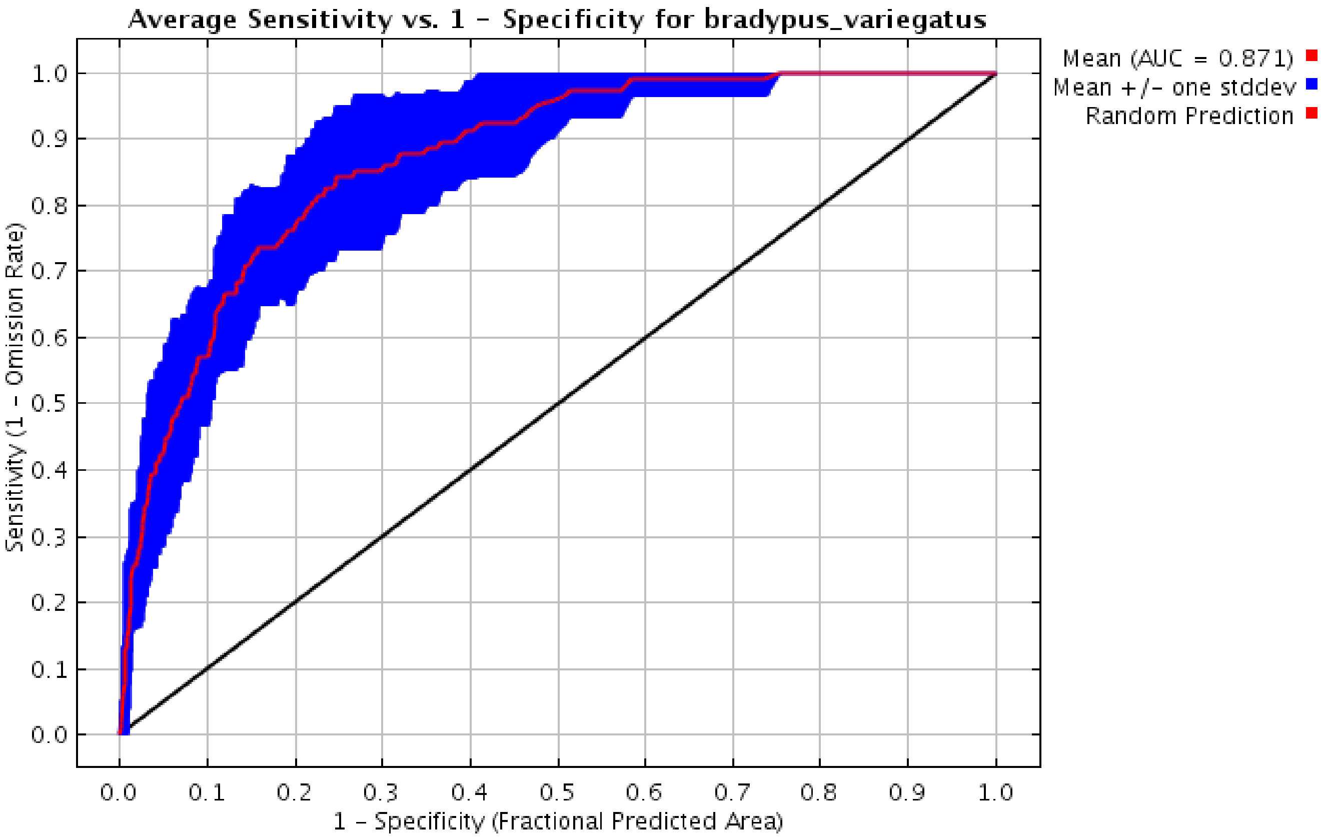
The effect is to tell the program where to find environmental layers and samples file and where to put outputs, to indicate that the ecoreg variable is categorical. The "autorun" flag tells the program to start running immediately, without waiting for the "Run" button to be pushed. Now try double clicking on the file to see what it does.

Many aspects of the Maxent program can be controlled by command-line arguments – press the "Help" button to see all the possibilities. Multiple runs can appear in the same file, and they will simply be run one after the other. You can change the default values of parameters by adding command-line arguments to the "maxent.bat" file. Many of the command-line arguments also have abbreviations, so the run described in batchExample.bat could also be initiated using this command:

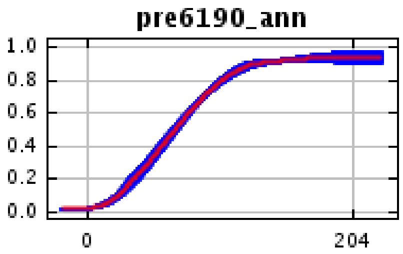
```
java -mx512m -jar maxent.jar –e layers –t eco –s samples\bradypus.csv –o outputs –r -a
```

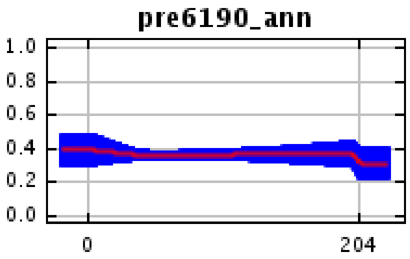
Replication

The "replicates" option can be used to do multiple runs for the same species. The most common uses for this flag are for repeated subsampling and for cross-validation. Replication can be controlled either from the Settings panel, or using command line arguments. By default, the form of replication used is cross-validation, where the occurrence data is randomly split into a number of equal-size groups called “folds”, and models are created leaving out each fold in turn. The left-out folds are then used for evaluation. Cross-validation has one big advantage over using a single training/test split: it uses all of the data for validation, thus making better use of small data sets. As an example, doing a run with the number of replicates set to 10 creates 10 html pages, plus a page that summarizes statistical information for the cross-validation. For example, we get ROC curves with error bars and average AUC across models, and summary response curves with one standard deviation error bars. For *Bradypus*, the cross-validated ROC curve shows some variability between models:



The single-variable response of *Bradypus* to annual precipitation shows little variation (on the left, below), while the marginal response to annual precipitation is more variable (below, right).



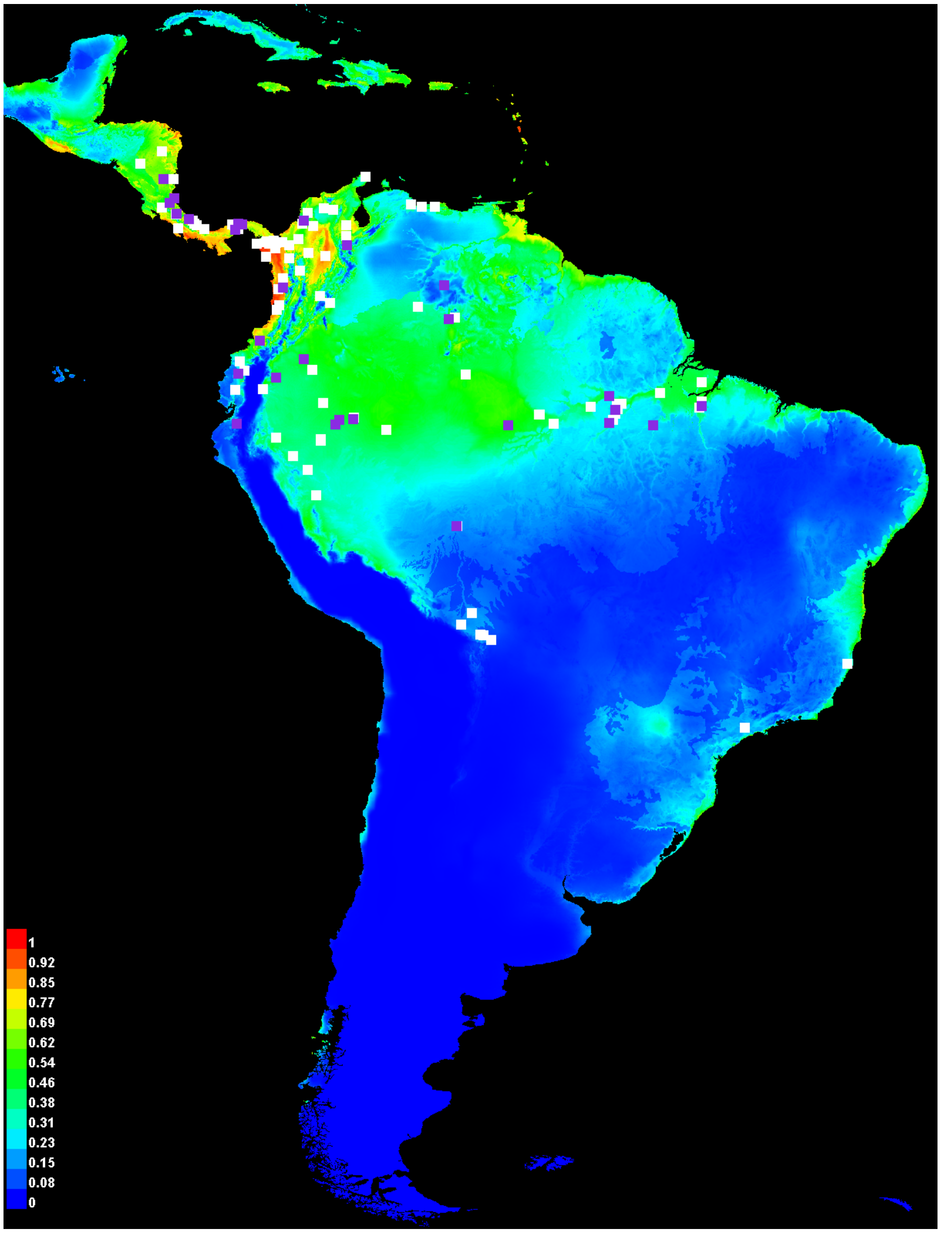


Two alternative forms of replication are supported: repeated subsampling, in which the presence points are repeatedly split into random training and testing subsets, and bootstrapping, where the training data is selected by sampling with replacement from the presence points, with the number of samples equaling the total number of presence points. With bootstrapping, the number of presence points in each set equals the total number of presence points, so the training data sets will have duplicate records.

With all three forms of replication, you may want to avoid eating up disk space by turning off the "write output grids" option, which will suppress writing of output grids for the replicate runs, so that you only get the summary statistics grids (avg, stderr etc.).

Regularization.

The “regularization multiplier” parameter on the settings panel affects how focused or closely-fitted the output distribution is – a smaller value than the default of 1.0 will result in a more localized output distribution that is a closer fit to the given presence records, but can result in to overfitting (fitting so close to the training data that the model doesn’t generalize well to independent test data). A larger regularization multiplier will give a more spread out, less localized prediction. Try changing the multiplier, and examine the pictures produced and changes in the AUC. As an example, setting the multiplier to 3 makes the following picture, showing a much more diffuse distribution than before:

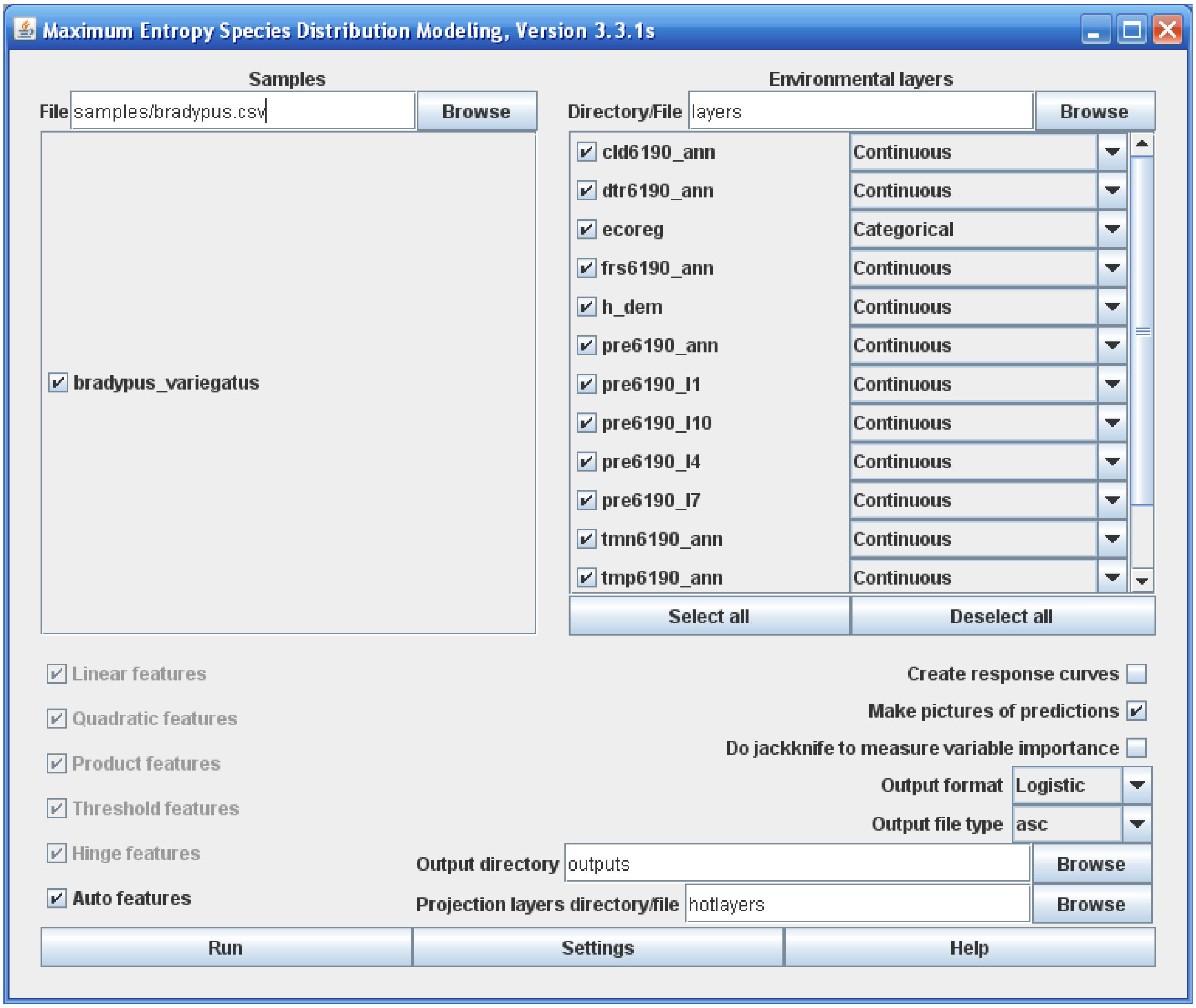


The potential for overfitting increases as the model complexity increases. First try setting the multiplier very small (e.g. 0.01) with the default set of features to see a highly overfit model. Then try the same regularization multiplier with only linear and quadratic features.

Projecting

A model trained on one set of environmental layers (or SWD file) can be "projected" by applying it to another set of environmental layers (or SWD file). Situations where projections are needed include modeling species distributions under changing climate conditions, applying a model of the native distribution of an invasive species to assess invasive risk in a different geographic area, or simply evaluating the model at a set of test locations in order to do further statistical analysis. Here we're going to use projection for a simplistic climate change prediction, and to give a taste of the difficulties involved in making reliable predictions of distributions under climate change.

The directory "hotlayers" has the same environmental data as the "layers" directory, with two changes: the annual average temperature variable (tmp6190_ann.asc) has all values increased by 30, representing a uniform 3 degree Celsius increase, while the maximum temperature variable (tmx6190_ann.asc) has all values increased by 40, representing a 4 degree Celsius increase. These changes represent a very simplified estimate of future climate, with higher average temperature and higher temperature variability, but with no change in precipitation. To apply a model of *Bradypus* to this new climate, enter the samples file and current environmental data as before, using either grids or SWD format, and enter the "hotlayers" directory in the "Projection Layers Directory", as pictured below.



Maximum Entropy Species Distribution Modeling, Version 3.3.1s

Samples

File samples/bradypus.csv

☑ bradypus_variegatus

Environmental layers

Directory/File layers

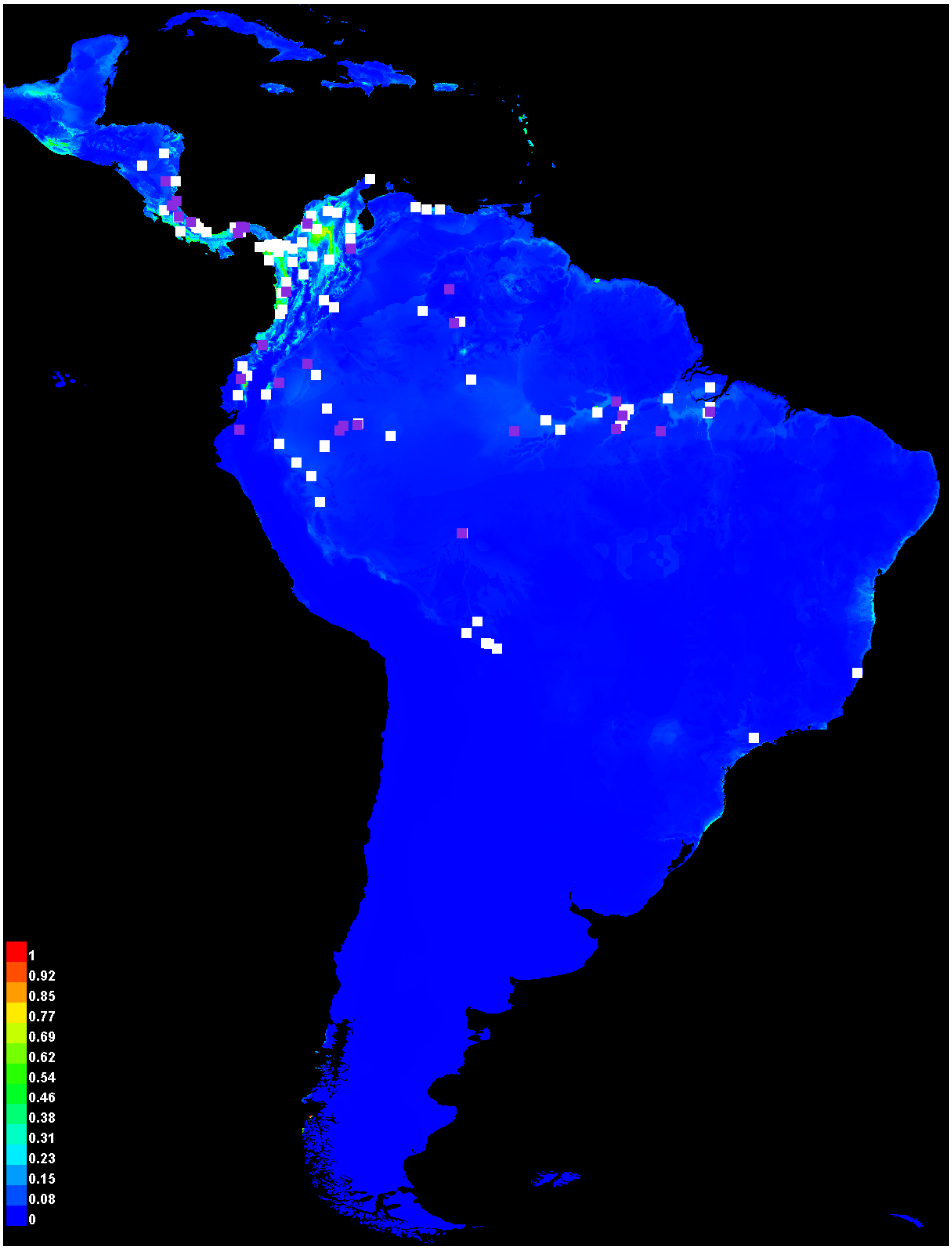
☑ cld6190_ann	Continuous
☑ dtr6190_ann	Continuous
☑ ecoreg	Categorical
☑ frs6190_ann	Continuous
☑ h_dem	Continuous
☑ pre6190_ann	Continuous
☑ pre6190_l1	Continuous
☑ pre6190_l10	Continuous
☑ pre6190_l4	Continuous
☑ pre6190_l7	Continuous
☑ tmn6190_ann	Continuous
☑ tmp6190_ann	Continuous

☑ Linear features
☑ Quadratic features
☑ Product features
☑ Threshold features
☑ Hinge features
☑ Auto features

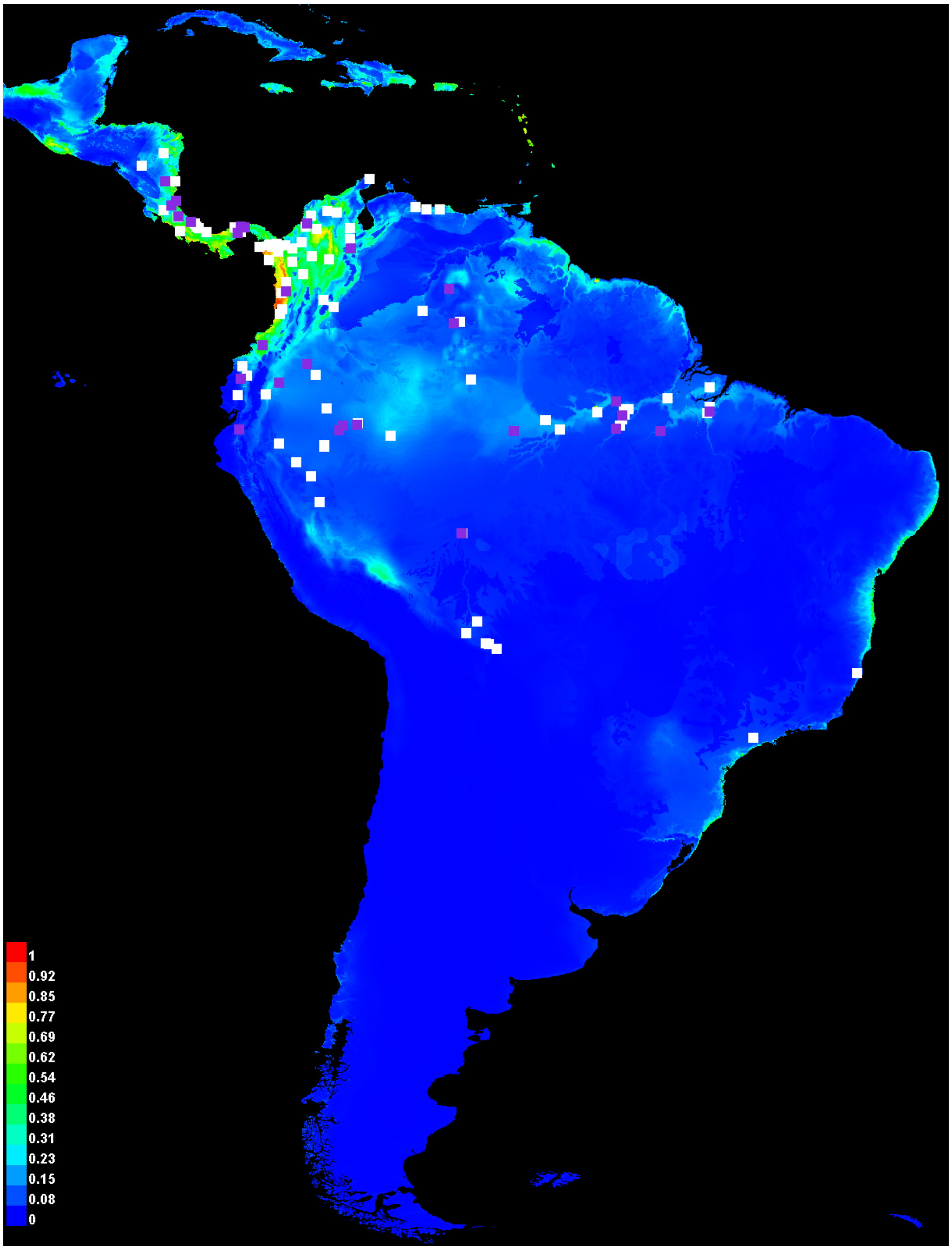
Create response curves □
Make pictures of predictions ☑
Do jackknife to measure variable importance □
Output format Logistic
Output file type asc
Output directory outputs
Projection layers directory/file hotlayers

The projection layers directory (or SWD file) must contain variables with the same names as the variables used for training the model, but describing a different conditions (e.g., a different geographic region or different climatic model). For both the training and projection data, each variable name is either the column title (if using an SWD format file) or the filename without the .asc file ending (if using a directory of grids).

When you press "Run", a model is trained on the environmental variables corresponding to current climate conditions, and then projected onto the ascii grids in the "hotlayers" directory. The output ascii grid is called "bradypus_variegatus_hotlayers.asc", and in general, the projection directory name is appended to the species name, in order to distinguish it from the standard (un-projected) output. If "make pictures of predictions" is selected, a picture of the projected model will appear in the "bradypus.html" file. In our case, this produces the following picture:

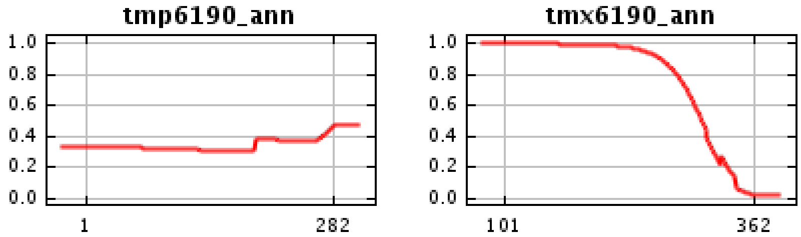


We see that the predicted probability of presence is drastically lower under the warmer climate. The prediction is of course dependent on the parameters of the model we're projecting. If we use only hinge and categorical features, rather than the default set of features, the projected distribution is more substantial:

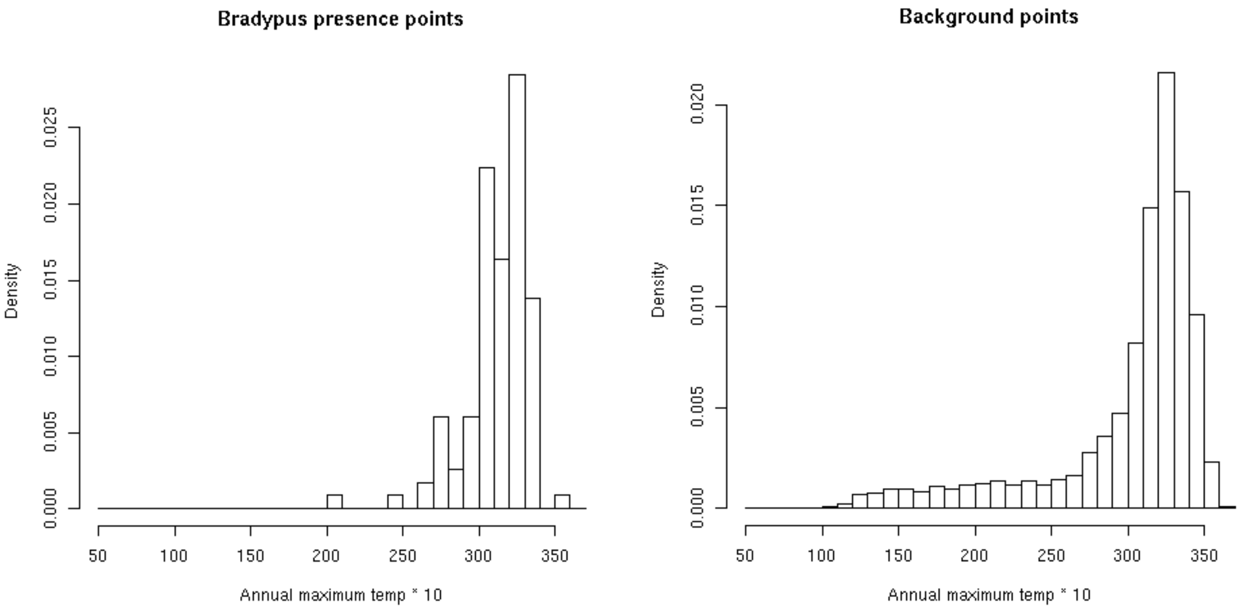


Two different models that look very similar in the area used for training may look very different when projected to a new geographic area or new climate conditions. This is especially true if there are correlated variables that allow a variety of ways to fit similar-looking models, since the correlations between the variables may change in the area you're projecting to.

Is the predicted range reduction of *Bradypus* under climate change reasonable? If we look at the marginal response curves for the model made with default features, we see that the maximum temperature is exerting a much stronger influence on the prediction:



Looking at a histogram of maximum temperature values at the known occurrences for *Bradypus*, we see that most occurrences (about 80%) have maximum temperature between 30 and 34 degrees Celsius. Only a single occurrence is above 34 degrees, even though a significant fraction of the background is between 34 and 35 degrees.

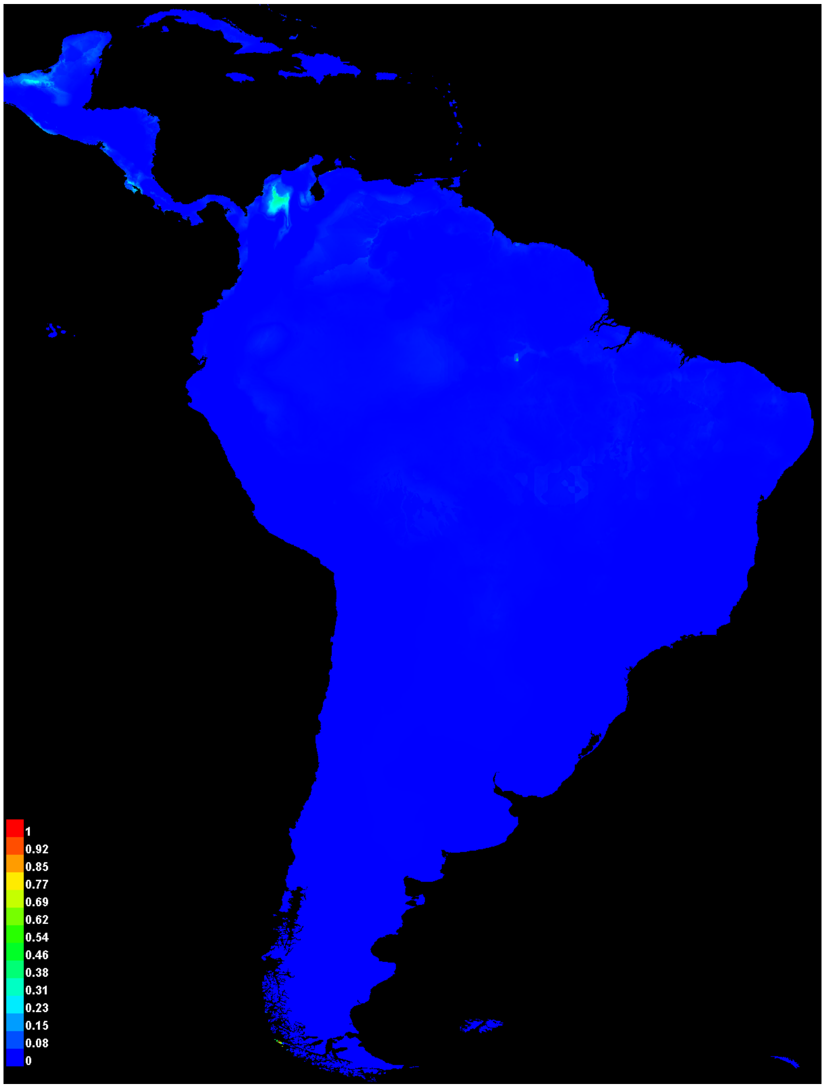


Under our climate change prediction, all 80% of the *Bradypus* locations currently above 30 degrees will have the maximum temperature increase to above 34 degrees. Therefore it may indeed be reasonable to predict that such locations will not be suitable for *Bradypus*, so *Bradypus* might not survive in most of its current range. Note that it is difficult to make any conclusions about why such conditions are not suitable: it may be that *Bradypus* is intolerant to heat, or it may be that higher maximum temperatures would allow fire to cause widespread replacement of rainforest by fire-tolerant tree species, eliminating most suitable habitat for *Bradypus*. To further investigate the prospects of *Bradypus* under climate change, we could do physiological studies to investigate the species' tolerance for heat, or study the fire ecology of rainforest boundaries in the region.

Note: histograms like the two above are useful tools for investigating your data. They were made in R using the following commands:

```
swdPresence <- read.csv("swd/bradypus_swd.csv")
hist(swdPresence$tmx6190_ann, probability=TRUE, breaks=c(5:37*10), xlab="Annual maximum temp * 10", main="Bradypus presence points")
swdBackground <- read.csv("swd/background.csv")
hist(swdBackground$tmx6190_ann, probability=TRUE, breaks=c(5:37*10), xlab="Annual maximum temp * 10", main="Background points")
```

We can see from the histograms that *Bradypus* can occasionally tolerate high temperatures, as evidenced by the single record with maximum temperature of 35 degrees. On the other had, there are extremely few points in the background with temperatures of 36 or above, so we have no evidence of whether or not *Bradypus* can tolerate even higher temperatures, which will be widespread under the future climate prediction. This is known as the problem of novel climate conditions: when projecting, the predictor variables may take on values outside the range seen during model training. The primary way Maxent deals with this problem is “clamping”, which treats variables outside the training range as if they were at the limit of the training range. This effect can be seen in the response curves described above, as the response is held constant outside the training range. Whenever a model is projected, Maxent makes a picture that shows where clamping has had a large effect. Projecting the *Bradypus* model made with all features gives this clamping picture, where the values depicted are the absolute difference between predictions with and without clamping.

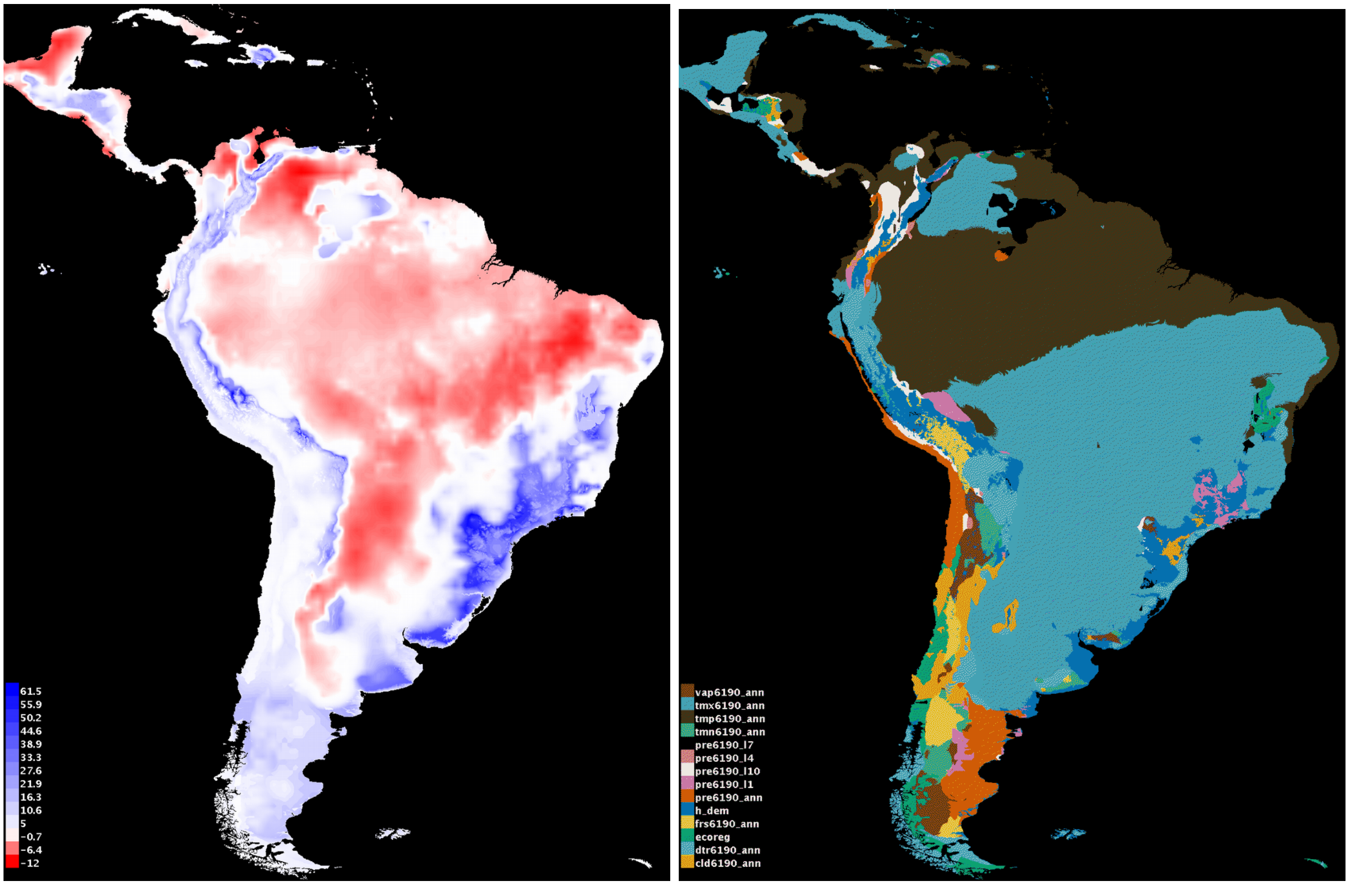


Clamping has clearly had little effect in this case – in particular, the response curve for maximum temperature above shows that the prediction had already leveled off near zero at the hot end of the scale, so clamping has little effect.

We can also compare the environmental variables used for projection to those used for training the model. Running the following batch command:

```
java -mx512m -cp maxent.jar density.tools.Novel layers hotlayers novel.asc
```

makes the following two pictures:



The leftmost picture shows how novel each point is in the *hotlayers* climate conditions. Negative values (shown in red) indicate novel climate, i.e., *hotlayers* values outside the range in *layers* – the value shown is the minimum over predictor variables of how far out of range the point is, expressed as a fraction of the range of that variable's values in *layers*. Positive values (shown in blue) are similar to BIOCLIM values, with a score of 100 meaning that a point is not at all novel, in the sense that its *hotlayers* values are all exactly equal to the median value in *layers*. The picture on the right shows which variable is most novel at each point, and it shows that novel climate conditions in *hotlayers* are due to average temperature (brown, mostly north of the Amazon River) or maximum temperature (teal blue, mostly south of the Amazan River).

Additional command-line tools

The Maxent jar file contains a number of tools that can be accessed from the command line. For Microsoft users: the features described here can be used in a batch file, like maxent.bat. As an alternative, Start->run->cmd gets you a shell for running commands interactively; cygwin (available free online) is a good alternative with a much more powerful shell that offers many unix utilities.

Quick visualization of grid file

Grid files in .asc, .grd and .mxe format, and some files in .bil format, can be viewed using the following command:

```
  java -mx512m -cp maxent.jar density.Show filename
```

As with all the commands described below, you may need to add the path to the maxent.jar file and/or the file you want to view. For example, you might use:

```
  java -mx1000m -cp C:\maxentfiles\maxent.jar density.Show C:\mydata\var1.asc
```

Show can take some optional arguments (immediately after density.Show):
 -s sampleFile   gives a file with presences to be shown in white dots
 -S speciesname   says which species in the sampleFile to show with dots
 -r radius   controls the size of the white and purple dots for occurrence records
 -L   removes the legend
 -o   writes the picture to a file in .png format

With a little Windows wizardry, you can make Show be invoked just by clicking on .asc, .grd or .mxe files. Make a batch file, say called showFile.bat, with the following single line in it:

```
  java -mx512m -cp "c:\maxentfiles\maxent.jar" density.Show %1
```

then associate files of type .asc, .grd or .mxe with the batch file: from a windows explorer (a.k.a. "My Computer"), Tools->Folder Options->File Types... You may need to make the batch file executable: right click on it and follow directions.

Making an SWD file

To make an SWD-format file from a non-SWD file:

```
  java -cp maxent.jar density.Getval samplesfile grid1 grid2 ...
```

where samplesfile is .csv file of occurrence data and grid1, grid2, etc. are grids in .asc, .mxe, .grd or .bil format. The output is written to "standard output", which means it appears in the command window. To write the output to a file, use a "redirect":

```
  java -cp maxent.jar density.Getval samplesfile grid1 grid2 ... > outfile
```

If all the grids are in a directory you can avoid having to list them all by name by using a "wildcard":

```
   java -cp maxent.jar density.Getval samplesfile directory/*.asc ... > outfile
```

because the wildcard (*) gets expanded to a list of all files that match.

Making an SWD background file

To pick a collection of background points uniformly at random from your study area:

```
   java -cp maxent.jar density.tools.RandomSample num grid1 grid2 ...
```

where "num" is the number of background points desired.

Calculating AUC

The following command:

```
   java -cp maxent.jar density.AUC testpointfile predictionfile
```

will calculate a presence-background AUC, where the presence points are given in the testpointfile and background points are drawn randomly from the predictionfile. The testpointfile is a .csv file (which may optionally be swd format), while the predictionfile is a grid file, typically representing the output of a species distribution model.

Projection

This tool allows you to apply a previously-calculated Maxent model to a new set of environmental data:

```
   java -cp maxent.jar density.Project lambdaFile gridDir outFile [args]
```

Here lambdaFile is a .lambdas file describing a Maxent model, and gridDir is a directory containing grids for all the predictor variables described in the .lambdas file. As an alternative, gridDir could be an swd format file. The optional args can contain any flags understood by Maxent -- for example, a "grd" flag would make the output grid of density.Project be in .grd format.

File conversion

To convert a directory full of grids in one format to another:

```
   java -cp maxent.jar density.Convert indir insuffix outdir outsuffix
```

where indir and outdir are directories and insuffix and outsuffix are one of asc, mxe, grd or bil.

Analyzing Maxent output in R

Maxent produces a number of output files for each run. Some of these files can be imported into other programs if you want to do your own analysis of the predictions. Here we demonstrate the use of the free statistical package R on Maxent outputs: this section is intended for users who have experience with R. We will use the following two files produced by Maxent:

```
  bradypus_variegatus_backgroundPredictions.csv
  bradypus_variegatus_samplePredictions.csv
```

The first file is only produced when the “writebackgroundpredictions” option is turned on, either by using a command-line flag or by selecting it from Maxent’s settings panel. The second file is always produced. Make sure you have test data (for example, by setting the random test percentage to 25); we will be evaluating the Maxent outputs using the same test data Maxent used. First we start R, and install some packages (assuming this is the first time we’re using them) and then load them by typing (or pasting):

```
  install.packages("ROCR", dependencies=TRUE)
  install.packages("vcd",  dependencies=TRUE)
  library(ROCR)
  library(vcd)
  library(boot)
```

Throughout this section we will use blue text to show R code and commands and green to show R outputs. Next we change directory to where the Maxent outputs are, for example:

```
 setwd("c:/maxent/tutorial/outputs")
```

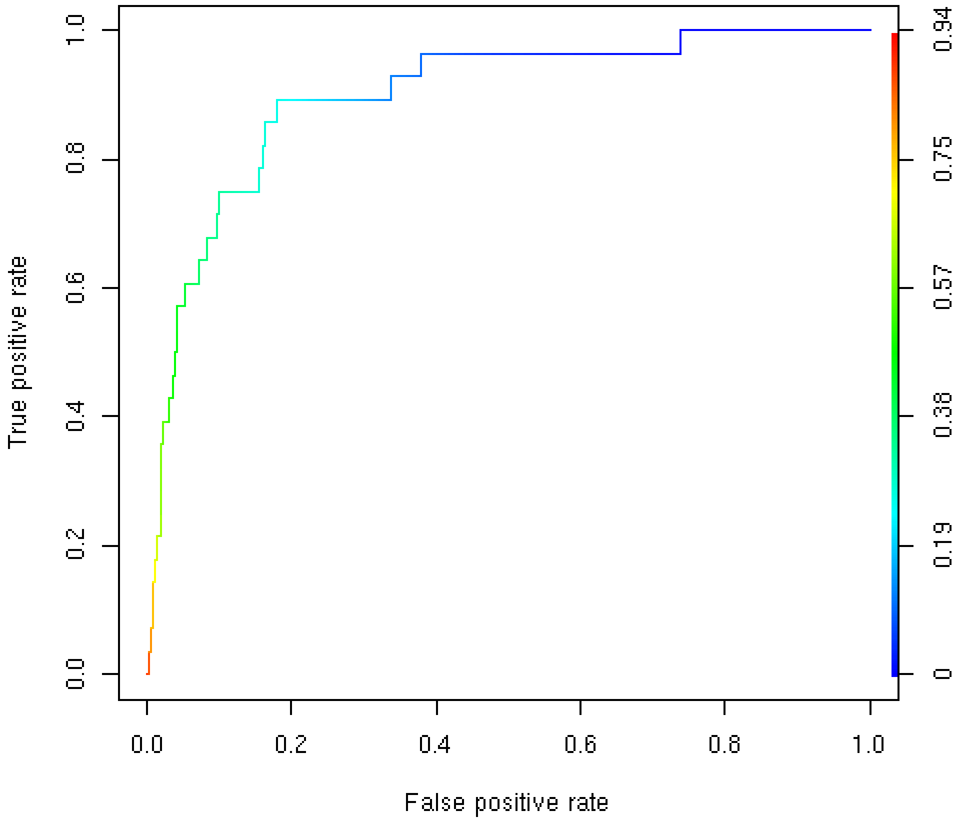
and then read in the Maxent predictions at the presence and background points, and extract the columns we need:

```
  presence <- read.csv("bradypus_variegatus_samplePredictions.csv")
  background <- read.csv("bradypus_variegatus_backgroundPredictions.csv")
  pp <- presence$Logistic.prediction                # get the column of predictions
  testpp <- pp[presence$Test.or.train=="test"]       # select only test points
  trainpp <- pp[presence$Test.or.train=="train"]   # select only test points
  bb <- background$logistic
```

Now we can put the prediction values into the format required by ROCR, the package we will use to do some ROC analysis, and generate the ROC curve:

```
  combined <- c(testpp, bb)                                 # combine into a single vector
  label <- c(rep(1,length(testpp)),rep(0,length(bb)))  # labels: 1=present, 0=random
  pred <- prediction(combined, label)                   # labeled predictions
  perf <- performance(pred, "tpr", "fpr")               # True / false positives, for ROC curve
  plot(perf, colorize=TRUE)                             # Show the ROC curve
  performance(pred, "auc")@y.values[[1]]                # Calculate the AUC
```

The plot command gives the following result:



while the “performance” command gives an AUC value of 0.8677759, consistent with the AUC reported by Maxent. Next, as an example of a test available in R but not in Maxent, we will make a bootstrap estimate of the standard deviation of the AUC.

```
 AUC <- function(p,ind) {
   pres <- p[ind]
   combined <- c(pres, bb)
   label <- c(rep(1,length(pres)),rep(0,length(bb)))
   predic <- prediction(combined, label)
   return(performance(predic, "auc")@y.values[[1]])
 }

b1 <- boot(testpp, AUC, 100)  # do 100 bootstrap AUC calculations
b1                            # gives estimates of standard error and bias
```

This gives the following output:

```
 ORDINARY NONPARAMETRIC BOOTSTRAP
```

```
  Call:
  boot(data = testpp, statistic = AUC, R = 100)

  Bootstrap Statistics :
      original      bias    std. error
t1* 0.8677759 -0.0003724138  0.02972513
```

and we see that the bootstrap estimate of standard error (0.02972513) is close to the standard error computed by Maxent (0.028). The bootstrap results can also be used to determine confidence intervals for the AUC:

```
  boot.ci(b1)
```

gives the following four estimates – see the resources section at the end of this tutorial for references that define and compare these estimates.

```
  Intervals :
  Level      Normal              Basic
  95%   ( 0.8099,  0.9264 )   ( 0.8104,  0.9291 )

  Level     Percentile            BCa
  95%   ( 0.8064,  0.9252 )   ( 0.7786,  0.9191 )
```

Those familiar with use of the bootstrap will notice that we are bootstrapping only the presence values here. We could also bootstrap the background values, but the results would not change much, given the very large number of background values (10000).

As a final example, we will investigate the calculation of binomial and Cohen's Kappa statistics for some example threshold rules. First, the following R code calculates Kappa for the threshold given by the minimum presence prediction:

```
  confusion <- function(thresh) {
    return(cbind(c(length(testpp[testpp>=thresh]), length(testpp[testpp<thresh])),
                 c(length(bb[bb>=thresh]), length(bb[bb<thresh]))))
  }
  mykappa <- function(thresh) {
    return(Kappa(confusion(thresh)))
  }
  mykappa(min(trainpp))
```

which gives a value of 0.0072. If we want to use the threshold that minimizes the sum of sensitivity and specificity on the test data, we can do the following, using the true positive rate and false positive rate values from the "performance" object used above to plot the ROC curve:

```
  fpr = perf@x.values[[1]]
  tpr = perf@y.values[[1]]
  sum = tpr + (1-fpr)
  index = which.max(sum)
  cutoff = perf@alpha.values[[1]][[index]]
  mykappa(cutoff)
```

This gives a kappa value of 0.0144. To determine binomial probabilities for these two threshold values, we can do:

```
  mybinomial <- function(thresh) {
    conf <- confusion(thresh)
    trials <- length(testpp)
    return(binom.test(conf[[1]][[1]], trials, conf[[1,2]] / length(bb), "greater"))
  }
  mybinomial(min(trainpp))
  mybinomial(cutoff)
```

This gives p-values of 5.979e-09 and 2.397e-11 respectively, which are both slightly larger than the p-values given by Maxent. The reason for the difference is that the number of test samples is greater than 25, the threshold above which Maxent uses a normal approximation to calculate binomial p-values.