Journal of Statistical Software

January 2008, Volume 23, Issue 11. *http://www.jstatsoft.org/*

PresenceAbsence: An R Package for Presence Absence Analysis

Elizabeth A. Freeman
USDA Forest Service

Gretchen Moisen
USDA Forest Service

Abstract

The **PresenceAbsence** package for R provides a set of functions useful when evaluating the results of presence-absence analysis, for example, models of species distribution or the analysis of diagnostic tests. The package provides a toolkit for selecting the optimal threshold for translating a probability surface into presence-absence maps specifically tailored to their intended use. The package includes functions for calculating threshold dependent measures such as confusion matrices, percent correctly classified (PCC), sensitivity, specificity, and Kappa, and produces plots of each measure as the threshold is varied. It also includes functions to plot the Receiver Operator Characteristic (ROC) curve and calculates the associated area under the curve (AUC), a threshold independent measure of model quality. Finally, the package computes optimal thresholds by multiple criteria, and plots these optimized thresholds on the graphs.

Keywords: binary classification, ROC, AUC, sensitivity, specificity, threshold, species distribution models, diagnostic tests.

1. Introduction

Binary classification is a technique crucial to multiple areas of study. In the medical field these techniques are used to evaluate diagnostic tests (Greiner *et al.* 2000; Cantor *et al.* 1999) and in radiology (Hanley and McNeil 1982). Ecological applications include predicting and mapping species distributions (Guisan and Thuiller 2005; Moisen *et al.* 2006; Anderson *et al.* 2003), conservation planning (Wilson *et al.* 2004; Fielding and Bell 1997), remote sensing (Congalton 1991), habitat modeling (Hirzel *et al.* 2006; Pearce and Ferrier 2000; Reineking and Schröder 2003), and wildlife biology (Manel *et al.* 1999; Guisan and Hofer 2003).

Many modeling techniques for binary classification problems result in predictions that are analogous to a probability of presence. To translate this to a simple 0/1 classification it is

necessary to specify a choice of threshold, or cut-off probability beyond which something is classified as present. The selection of this threshold value can have dramatic effects on model accuracy and predicted prevalence (the overall proportion of locations where the variable is predicted to be present). The traditional default is to simply use a threshold of 0.5 as the cutoff, but this does not necessarily preserve the prevalence or result in the highest prediction accuracy, especially for data sets with very high or very low prevalence.

The **PresenceAbsence** package (Freeman 2007) in R (R Development Core Team 2007) is a collection of tools (both analytical and graphical) for evaluating the performance of binary classification models and determining the optimum threshold for translating a probability surface to a presence-absence map. The **PresenceAbsence** package includes functions for calculating threshold dependent accuracy measures such as confusion matrices, percent correctly classified (PCC), sensitivity, specificity, and Kappa, and produces several types of graphical plots illustrating the effect on each measure as the threshold is varied. The package calculates optimal thresholds by multiple criteria, and plots these optimized thresholds on the graphs. It also includes functions to plot the Receiver Operator Characteristic (ROC) curve and calculates the associated area under the curve (AUC), a threshold independent measure of model quality (Hanley and McNeil 1982; Swets 1988). The package produces calibration plots to assess the accuracy of the predicted probabilities. Finally, it includes a function that uses beta distributions to produce simulated presence-absence data.

There are stand alone software applications for evaluating binary classification models. Examples range from full featured ROC analysis tools such as **ROC_AUC** (Schröder 2006), to general purpose model evaluation tools such as **PERF** (Caruana 2004), to the Web-based ROC calculators **ROCOn** (Farrand and Langford 2002) and **jrocfit.org** (Eng 2006). In R, it is possible to calculate the AUC from the Wilcox test provided in the **stats** package, part of a standard installation of R. There are also several R packages available that will produce ROC plots and calculate performance measures such as the AUC. These include the packages **ROCR** (Sing *et al.* 2005) and the **ROC** (Carey and Redestig 2007), as well as the `roc.area` and `roc.plot` functions from **verification** package (NCAR - Research Application Program 2007), the `colAUC` function from the **caTools** package (Tuszynski 2007), the `ROC` function from the **Epi** package (Carstensen *et al.* 2007), and the `auROC` function from the **limma** package (Smyth 2005). Some of these R packages will optimize the threshold choice by one or more techniques and plot this threshold on the ROC plot, or produce plots of sensitivity and specificity as a function of threshold. In S-PLUS (Insightful Corp. 2003), the **roc** package (Atkinson and Mahoney 2004) goes beyond the **PresenceAbsence** package by providing significance testing of correlated ROC curves.

The **PresenceAbsence** package, however, provides multiple graph types, model evaluation statistics, and threshold optimization criteria in a single integrated unit. It allows researchers to carry out a complete analysis of their model results, and immediately see the effects of their threshold selections on all aspects of the prediction accuracy. Thresholds can be optimized by 12 different techniques, and these optimized thresholds can be plotted automatically on the graphs. It also provides tools, both analytical and graphical, to examine the relationships between prevalence, model accuracy, and threshold selection.

If the model building is being carried out in R, then using the **PresenceAbsence** package (rather than stand alone ROC software) streamlines the model evaluation process. And if the final presentation of the results is created as a **Sweave** document, the entire analysis and document creation can be carried out in a single step, allowing for automatic updates of the

Symbol	Latin Name	Common Name
ABCO	Abies concolor	white fir
ABLA	Abies lasiocarpa	subalpine fir
ACGR3	Acer grandidentatum	bigtooth maple
CELE3	Cercocarpus ledifolius	curlleaf mountain-mahogany
JUOS	Juniperus osteosperma	Utah juniper
JUSC2	Juniperus scopulorum	Rocky Mountain juniper
PICO	Pinus contorta	lodgepole pine
PIED	Pinus edulis	common or twoneedle pinyon
PIEN	Picea engelmannii	Englemann spruce
PIPO	Pinus ponderosa	Ponderosa pine
POTR5	Populus tremuloides	quaking aspen
PSME	Pseudotsuga menziesii	Douglas-fir
QUGA	Quercus gambelii	Gambel oak

Table 1: Species codes for the 13 tree species.

figures for new data or slight changes in analysis.

In this paper, we give a demonstration of a basic exploratory analysis of a presence-absence dataset using this package. The **PresenceAbsence** package is available from the Comprehensive R Archive Network (http://CRAN.R-project.org/).

2. Dataset

This demonstration dataset is included in the **PresenceAbsence** package. It is the validation dataset from Moisen *et al.* (2006), and more detailed information can be found in this paper. In brief, presence-absence data for 13 tree species (Table 1) were collected by the USDA Forest Service, Forest Inventory and Analysis Program in a six million ha study area predominately in the mountains of northern Utah. These species presence data were modeled as functions of satellite imagery and bioclimatic information using three different types of models. These models included generalized additive models implemented in R, a variant on classification and regression trees implemented in Rulequest's **See5** package, and stochastic gradient boosting implemented in R, using the **gbm** package (Ridgeway 2006) (hereafter GAM, See5, and SGB, respectively). The demonstration data used here contains presence-absence predictions for the 13 tree species at 386 independent model validation sites. Specifically, the data set consists of species, observed presence-absence values (1 or 0), and the predicted probability of presence from each of the three modeling techniques.

After installing the **PresenceAbsence** package, load the dataset for this demo:

```
R> library("PresenceAbsence")
R> data("SPDATA")
```

Begin by examining the data structure:

```
R> head(SPDATA)
```

```
  SPECIES OBSERVED        GAM   See5        SGB
1    ABCO        0 0.03072217 0.0001 0.02615349
2    ABCO        0 0.09418715 0.0600 0.03536900
3    ABCO        0 0.00010000 0.0001 0.05954248
4    ABCO        0 0.08465268 0.1700 0.04357160
5    ABCO        1 0.69945011 0.2900 0.40515515
6    ABCO        0 0.99999111 0.1700 0.12727683
```

Column 1 is for the row ID's. In this case, The ID column is made up by the species code of each observation. Column 2 is the observed values where `0` = absent and `1` = present. If the observed values had been actual values (for example, basal area or tree counts) any values other than zero would have been treated by the functions as present. The remaining columns are for model predictions. Preferably these will be predicted probabilities ranging from zero to one. If all that is available are predicted presence-absence values, basic accuracy statistics can still be calculated, but you loose the ability to examine the effect of threshold choice, and most of the graphs are not meaningful. In this dataset, these are the model predictions from the three types of models: GAM, See5, and SGB.

Note that the functions rely on column order to identify the observed and predicted values, not on the column names. Therefore, as long as the columns are in the correct order, column names can be anything you choose. The only time the functions use the column names is for the default labels on graphs.

We will define a few variables for later use:

```
R> species <- as.character(unique(SPDATA$SPECIES))
R> model.names <- as.character(names(SPDATA)[-c(1, 2)])
R> N.models <- ncol(SPDATA) - 2
R> N.sp <- length(species)
R> N.obs <- length(SPDATA$SPECIES[SPDATA$SPECIES == species[1]])
R> Obs.prev <- table(SPDATA$SPECIES, SPDATA$OBSERVED)[, 2]/N.obs
R> Obs.prev <- round(Obs.prev, digits = 2)
```

3. Initial exploration

The `presence.absence.hist` function produces a bar plot of observed values as a function predicted probability. These plots make prevalence dramatically obvious. Many of the traditional accuracy measures (such as PCC, sensitivity and specificity) are highly dependent on species prevalence. Therefore, it is important to check prevalence early in an investigation.

Consider three models for a single species (Figure 1):

```
R> par(oma = c(0, 0, 3, 0), mar = c(4, 4, 4, 1), mfrow = c(1, 3), cex = 1)
R> sp <- 1
R> DATA <- SPDATA[SPDATA$SPECIES == species[sp], ]
R> for (mod in 1:N.models) {
+     presence.absence.hist(DATA, which.model = mod, legend.cex = 1,
+         N.bars = 20)
```

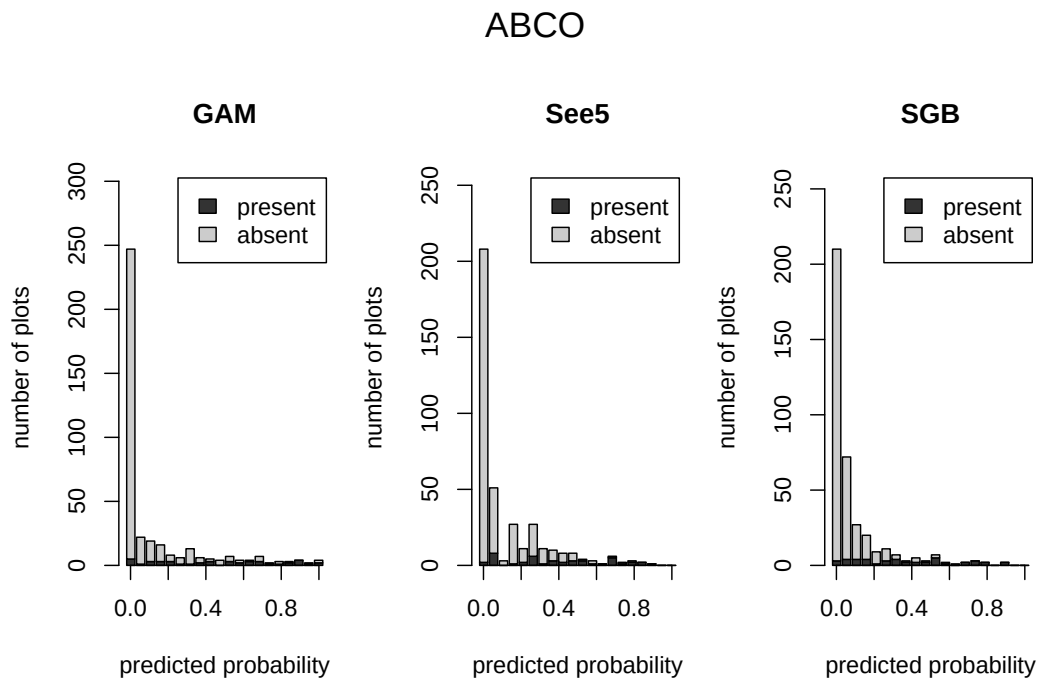

Figure 1: Presence-Absence histogram for three models and one species.

```
+ }
R> mtext(species[sp], side = 3, line = 0.5, cex = 1.5, outer = TRUE)
```

It is immediately obvious that this is a low prevalence species. This is important to note as extra care must be used when analyzing species with very low or very high prevalence. Many of the traditional measures of accuracy, such as percent correctly classified (PCC), sensitivity, and specificity are highly dependent on prevalence. Even some of the threshold optimization criteria can be affected by species prevalence (Manel *et al.* 2001).

The immediate effect of low prevalence on this particular plot is that the zero bar is so much taller than the other bars that it is difficult to see any other details of the species distributions. Setting the argument `truncate.tallest = TRUE` will truncate the tallest bar to fit better in the plot region and allow examination of finer structures in the histogram. If `truncate.tallest = TRUE` and the tallest bar is more than twice the height of the other bars, it is truncated so that it is 20 percent taller than the next bar, the truncated bar is cross hatched, and the true count is printed at the top of the bar. Note though, that the `truncate.tallest` option can visually obscure the true prevalence (Figure 2).

```
R> par(oma = c(0, 0, 3, 0), mar = c(4, 4, 4, 1), mfrow = c(1, 3), cex = 1)
R> sp <- 1
R> DATA <- SPDATA[SPDATA$SPECIES == species[sp], ]
R> for (mod in 1:N.models) {
+     presence.absence.hist(DATA, which.model = mod, legend.cex = 1,
```

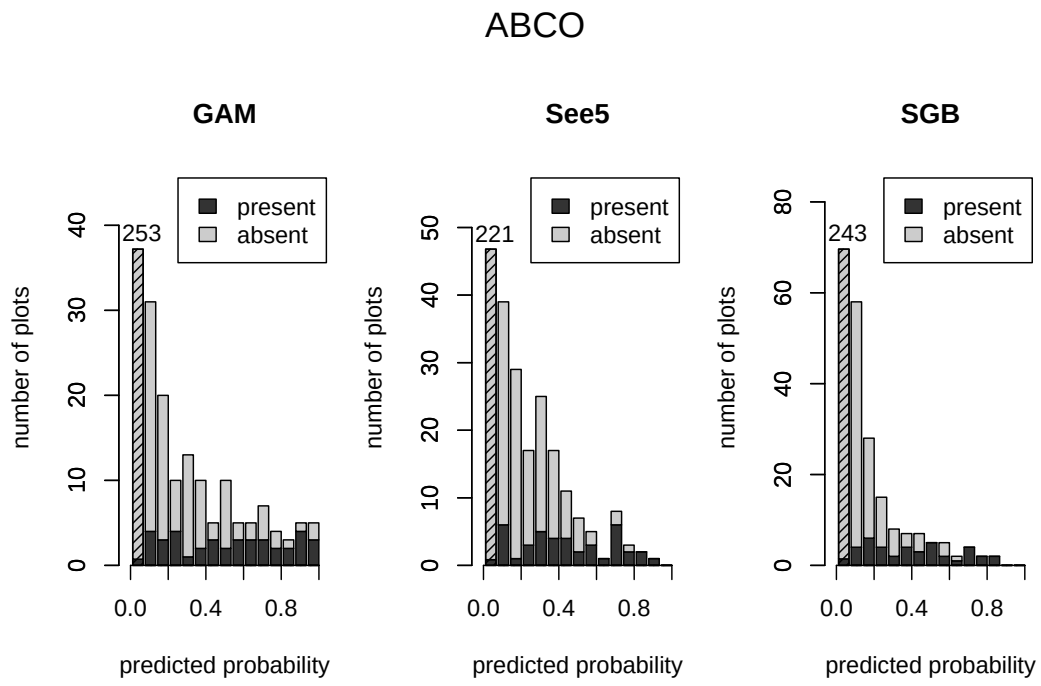

Figure 2: Presence-Absence histogram for three models and one species, with tallest bar truncated.

```
+         N.bars = 15, truncate.tallest = TRUE)
+ }

[1] "height of tallest bar truncated to fit on plot"
[1] "height of tallest bar truncated to fit on plot"
[1] "height of tallest bar truncated to fit on plot"

R> mtext(species[sp], side = 3, line = 0.5, cex = 1.5, outer = TRUE)
```

Next, we will compare the graphs of 3 species with differing prevalence (in this case the `truncate.tallest` argument is set to `FALSE` to avoid obscuring the true prevalence, see Figure 3):

```
R> par(oma = c(0, 0, 3, 0), mar = c(4, 4, 4, 1), mfrow = c(1, 3), cex = 1)
R> mod <- 2
R> for (sp in c("ACGR3", "PIEN", "POTR5")) {
+     presence.absence.hist(DATA = SPDATA[SPDATA$SPECIES ==
+         sp, ], which.model = mod, legend.cex = 1, N.bars = 15,
+         main = sp)
+ }
R> mtext(model.names[mod], side = 3, line = 0, cex = 1.5, outer = TRUE)
```

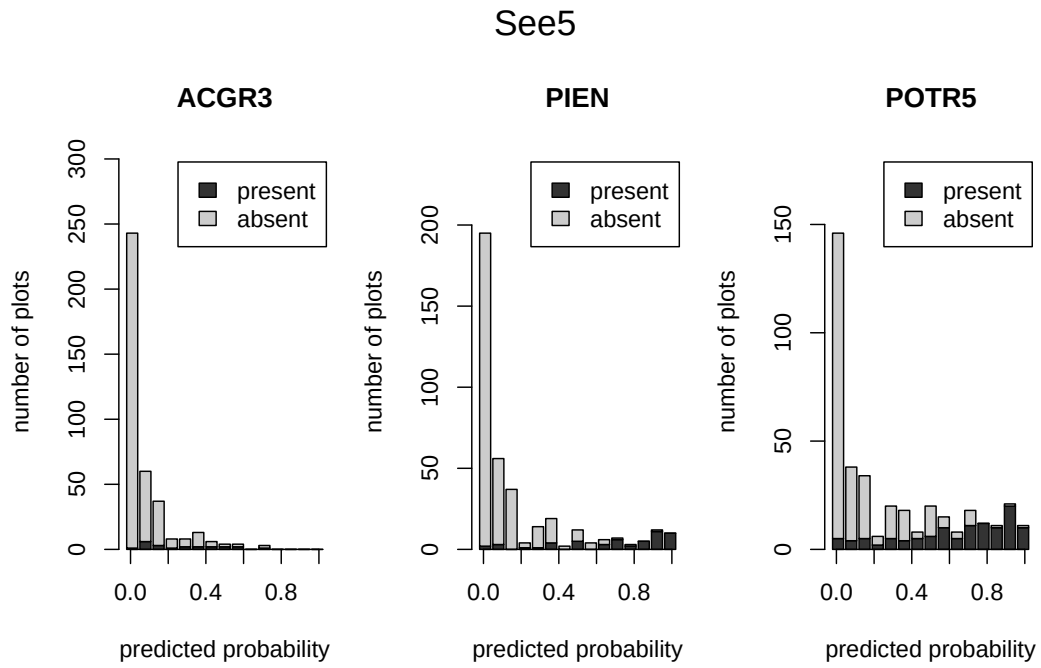

Figure 3: Presence-Absence histogram for the See5 model for three species with increasing prevalence.

These plots are also a good visual indicator of model quality. A more accurate model (such as the See5 model for PICO) will have a distinctly double humped histogram. Choosing a threshold in the valley will divide the data cleanly into observed present and observed absent groups. A less accurate model (such as the See5 model for JUSC2) will have considerable overlap between observed present and observed absent, and no threshold exists that will cleanly divide the data into observe present and observed absent. The challenge here is to find a threshold that offers the best compromise. There are several methods available to optimize the choice of threshold, but this will be dealt with in a later section.

Compare the histograms for the See5 models for three species with similar prevalence, but increasing model quality (Figure 4):

```
R> par(oma = c(0, 0, 3, 0), mar = c(4, 4, 4, 1), mfrow = c(1, 3), cex = 1)
R> mod <- 2
R> for (sp in c("JUSC2", "ABCO", "PICO")) {
+     presence.absence.hist(DATA = SPDATA[SPDATA$SPECIES ==
+         sp, ], which.model = mod, legend.cex = 1, N.bars = 15,
+         truncate.tallest = TRUE, main = sp)
+ }

[1] "height of tallest bar truncated to fit on plot"
[1] "height of tallest bar truncated to fit on plot"
[1] "height of tallest bar truncated to fit on plot"
```

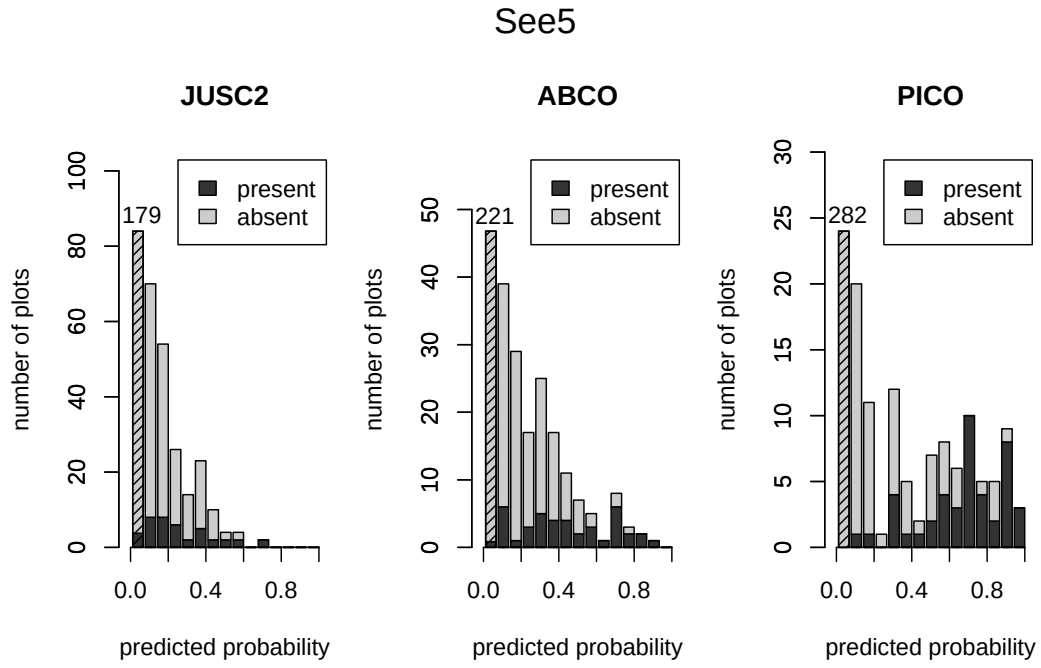

Figure 4: Presence-Absence histogram for the See5 model for three species with similar prevalence but increasing model quality.

```
R> mtext(model.names[mod], side = 3, line = 0, cex = 1.5, outer = TRUE)
```

4. Tables and calculations

4.1. Predicted prevalence

If it is important that the model predicts the prevalence of a species correctly, you can choose a threshold where the predicted prevalence equals the prevalence threshold. The function `predicted.prevalence` calculates a table showing the effect of threshold choice on the predicted prevalence of the models.

Begin with the standard cutoff of threshold = 0.5, for declaring a species present or absent:

```
R> DATA <- SPDATA[SPDATA$SPECIES == species[1], ]
R> pred.prev <- predicted.prevalence(DATA = DATA)
R> for (sp in 2:N.sp) {
+     DATA <- SPDATA[SPDATA$SPECIES == species[sp], ]
+     pred.prev <- rbind(pred.prev, predicted.prevalence(DATA = DATA))
+ }
R> pred.prev <- cbind(species = species, pred.prev)
```

```
R> pred.prev[, 3:6] <- round(pred.prev[, 3:6], digits = 2)
R> pred.prev

    species threshold Obs.Prevalence  GAM See5  SGB
1      ABCO       0.5           0.11 0.10 0.06 0.05
11     ABLA       0.5           0.19 0.17 0.18 0.15
12    ACGR3       0.5           0.06 0.08 0.02 0.02
13    CELE3       0.5           0.08 0.08 0.03 0.01
14     JUOS       0.5           0.27 0.26 0.26 0.25
15    JUSC2       0.5           0.12 0.05 0.02 0.01
16     PICO       0.5           0.11 0.13 0.13 0.10
17     PIED       0.5           0.24 0.24 0.26 0.24
18     PIEN       0.5           0.14 0.14 0.15 0.12
19     PIPO       0.5           0.10 0.09 0.07 0.07
110   POTR5       0.5           0.30 0.25 0.28 0.25
111    PSME       0.5           0.23 0.16 0.11 0.13
112    QUGA       0.5           0.15 0.14 0.08 0.10
```

We can look even closer at individual species to examine the effect of threshold choice on prevalence:

```
R> sp <- 1
R> DATA <- SPDATA[SPDATA$SPECIES == species[1], ]
R> pred.prev <- predicted.prevalence(DATA = DATA, threshold = 11)
R> pred.prev[, 2:5] <- round(pred.prev[, 2:5], digits = 2)
R> print(paste("Species:", species[sp]))

[1] "Species: ABCO"

R> pred.prev

   threshold Obs.Prevalence  GAM See5  SGB
1        0.0           0.11 1.00 1.00 1.00
2        0.1           0.11 0.30 0.33 0.27
3        0.2           0.11 0.21 0.25 0.15
4        0.3           0.11 0.18 0.15 0.10
5        0.4           0.11 0.13 0.10 0.07
6        0.5           0.11 0.10 0.06 0.05
7        0.6           0.11 0.08 0.04 0.03
8        0.7           0.11 0.05 0.02 0.02
9        0.8           0.11 0.03 0.01 0.01
10       0.9           0.11 0.02 0.00 0.00
11       1.0           0.11 0.00 0.00 0.00
```

Setting `threshold = 11` causes the function to calculate 11 evenly spaced thresholds between zero and one. If it is important that the model predicts the prevalence of a species correctly,

you can use this table to choose a threshold where the predicted prevalence equals the observed prevalence. To calculate the prevalence at particular thresholds, the `threshold` argument can be set to the values of interest, for example `threshold = c(0.23,0.6,0.97)`

4.2. Accuracy tables

The function `presence.absence.accuracy` calculates basic accuracy measures (PCC, sensitivity, specificity, Kappa, and AUC) for presence-absence data, and (optionally) the associated standard deviations. The **PresenceAbsence** package uses the method from DeLong *et al.* (1988) to calculate AUC and its associated standard deviation. By default the function will calculate all basic accuracy measures at the threshold of 0.5:

```
R> sp <- 1
R> DATA <- SPDATA[SPDATA$SPECIES == species[sp], ]
R> accu <- presence.absence.accuracy(DATA)
R> accu[, -c(1, 2)] <- signif(accu[, -c(1, 2)], digits = 2)
R> print(paste("Species:", species[sp]))

[1] "Species: ABCO"

R> accu

  model threshold  PCC sensitivity specificity Kappa  AUC PCC.sd
1   GAM       0.5 0.90        0.50        0.95  0.47 0.88  0.016
2  See5       0.5 0.91        0.36        0.98  0.44 0.86  0.014
3   SGB       0.5 0.91        0.34        0.99  0.44 0.91  0.014
  sensitivity.sd specificity.sd Kappa.sd AUC.sd
1          0.076         0.0120    0.072  0.025
2          0.073         0.0071    0.078  0.028
3          0.072         0.0058    0.079  0.023
```

The standard deviations can be suppressed by setting `st.dev=FALSE`. Note that the standard deviations calculated by `presence.absence.accuracy` are only appropriate for examining each model individually. To do significance testing for differences between models it is necessary to take into account correlation. DeLong *et al.* (1988) examines this issue and describes a method for comparing the AUC from two or more correlated models. This method is not included in the **PresenceAbsence** package, but can be found in the S-PLUS **roc** library from the Mayo clinic (Atkinson and Mahoney 2004). See `help(auc)` for further details.

While the default for `presence.absence.accuracy` is to calculate accuracy measures for all models at `threshold = 0.5`, the `threshold` argument can be used to set other thresholds. When the table is being produced for multiple models, `threshold` must be either a single threshold used for all models, or it must be a vector of thresholds of the same length as the number of models in the table. In the later case, the first threshold will be used for the first model, the second threshold for the second row, and so on. On the other hand, when examining a single model (the `which.model` argument can be used to specify the model), multiple thresholds can be calculated. This allows us to discover how the accuracy measures

change as a function of threshold. Thus when producing a table for a single model, the `threshold` argument can be given as a single threshold between zero and one, a vector of thresholds between zero and one, or a positive integer representing the number of evenly spaced thresholds to calculate.

```
R> sp <- 1
R> DATA <- SPDATA[SPDATA$SPECIES == species[sp], ]
R> accu <- presence.absence.accuracy(DATA, which.model = 3, threshold = 11,
+     st.dev = FALSE)
R> accu[, -c(1, 2)] <- signif(accu[, -c(1, 2)], digits = 2)
R> print(paste("Species:", species[sp]))

[1] "Species: ABCO"

R> accu

   model threshold  PCC sensitivity specificity Kappa  AUC
1    SGB       0.0 0.11       1.000        0.00 0.000 0.91
2    SGB       0.1 0.81       0.840        0.80 0.400 0.91
3    SGB       0.2 0.89       0.660        0.92 0.510 0.91
4    SGB       0.3 0.92       0.570        0.96 0.570 0.91
5    SGB       0.4 0.91       0.430        0.98 0.490 0.91
6    SGB       0.5 0.91       0.340        0.99 0.440 0.91
7    SGB       0.6 0.91       0.200        1.00 0.300 0.91
8    SGB       0.7 0.90       0.160        1.00 0.250 0.91
9    SGB       0.8 0.89       0.045        1.00 0.078 0.91
10   SGB       0.9 0.89       0.000        1.00 0.000 0.91
11   SGB       1.0 0.89       0.000        1.00 0.000 0.91

R> accu <- presence.absence.accuracy(DATA, which.model = 3,
+     threshold = c(0.2, 0.4, 0.5, 0.89), st.dev = FALSE)
R> accu[, -c(1, 2)] <- signif(accu[, -c(1, 2)], digits = 2)
R> print(paste("Species:", species[sp]))

[1] "Species: ABCO"

R> accu

  model threshold  PCC sensitivity specificity Kappa  AUC
1   SGB      0.20 0.89        0.66        0.92  0.51 0.91
2   SGB      0.40 0.91        0.43        0.98  0.49 0.91
3   SGB      0.50 0.91        0.34        0.99  0.44 0.91
4   SGB      0.89 0.89        0.00        1.00  0.00 0.91
```

In these tables, the AUC is constant for all thresholds. The AUC is the only one of these basic accuracy statistics that is threshold independent. Calculating the AUC for large datasets can

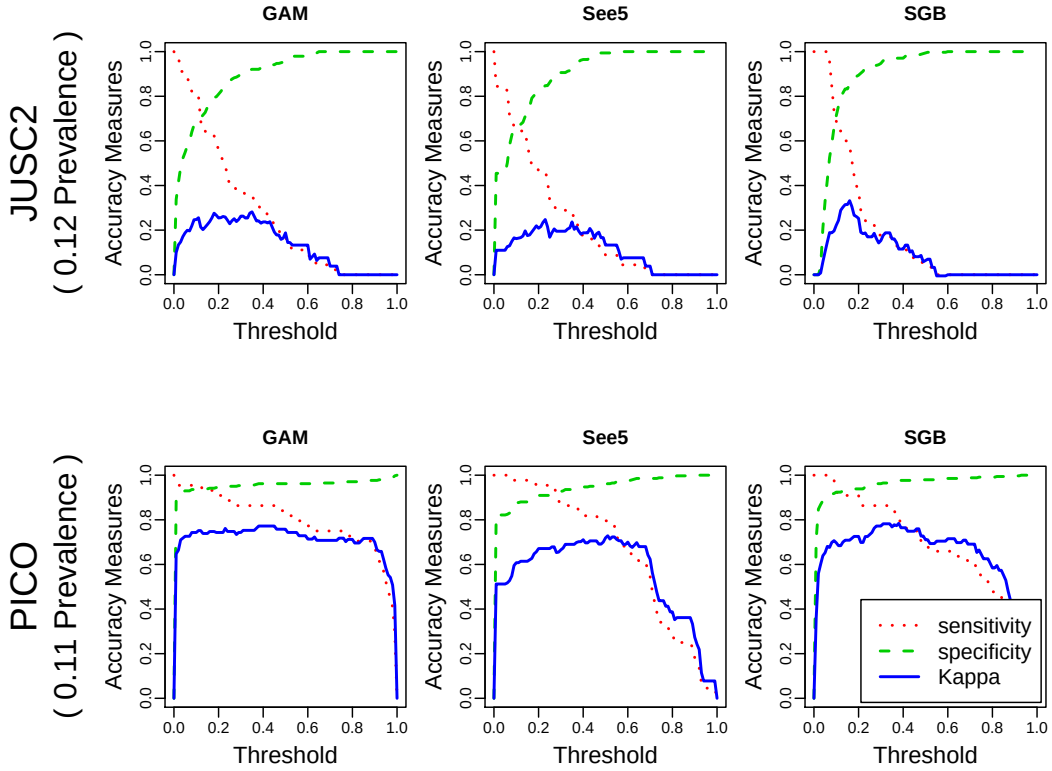

Figure 5: Graph of error measures (sensitivity, specificity, and Kappa) as a function of threshold for two species with similar prevalence but differing model quality.

be very memory intensive. Setting the argument `find.auc = FALSE` will turn off the AUC calculations, and increase the speed.

5. Graphing the error statistics as a function of threshold

Tables are useful for looking up specific values, but often it is easier to spot patterns visually in a graph. The **PresenceAbsence** package provides visual displays of error statistics.

In this dataset, JUSC2 and PICO have the same prevalence, but PICO has better model quality for all three models. This can be seen in these graphs in several ways. For PICO, Kappa reaches a higher maximum value, and Kappa stays at higher values for a greater range of possible thresholds. Also, the lines representing sensitivity and specificity cross at higher value, thus it is possible to choose a threshold that gives a high level of sensitivity without sacrificing specificity, and vice versa (Figure 5).

```
R> par(oma = c(0, 5, 0, 0), mar = c(3, 3, 3, 1), mfrow = c(2, 3),
+     cex = 0.7, cex.lab = 1.4, mgp = c(2, 0.5, 0))
R> sp <- c("JUSC2", "PICO")
```

```
R> for (i in 1:2) {
+     DATA = SPDATA[SPDATA$SPECIES == sp[i], ]
+     for (mod in 1:3) {
+         if (i == 2 & mod == 3) {
+             legend.flag <- TRUE
+         }
+         else {
+             legend.flag <- FALSE
+         }
+         error.threshold.plot(DATA, which.model = mod, color = TRUE,
+             add.legend = legend.flag, legend.cex = 1.3)
+         if (mod == 1) {
+             mtext(sp[i], side = 2, line = 6.3, cex = 1.6)
+             mtext(paste("(", Obs.prev[names(Obs.prev) == sp[i]],
+                 "Prevalence )"), side = 2, line = 4.3, cex = 1.3)
+         }
+     }
+ }
```

6. ROC plots and the AUC

ROC plots, with their associated AUC provide a threshold independent method of assessing model performance. ROC plots are produced by assessing the ratio of true positives to false positives for all possible thresholds. The AUC is equivalent to the chance that a randomly chosen plot with an observed value of present will have a predicted probability higher than that of a randomly chosen plot with an observed value of absent.

The plot from a good model will rise steeply to the upper left corner then level off quickly, resulting in an AUC near 1.0. A poor model (i.e. a model that is no better than random assignment) will have a ROC plot lying along the diagonal, with an AUC near 0.5 (Figure 6).

```
R> par(oma = c(0, 0, 0, 0), mfrow = c(1, 2), cex = 0.7, cex.lab = 1.5)
R> sp <- c("ACGR3", "POTR5")
R> for (i in 1:2) {
+     DATA <- SPDATA[SPDATA$SPECIES == sp[i], ]
+     auc.roc.plot(DATA, color = TRUE, legend.cex = 1.4, main = "")
+     mtext(sp[i], side = 3, line = 2.5, cex = 1.6)
+     mtext(paste(Obs.prev[names(Obs.prev) == sp[i]], "Prevalence"),
+         side = 3, line = 0.5, cex = 1.3)
+ }
```

Thresholds are not evenly spaced along the ROC plot, therefore it can be helpful to see where individual thresholds lie along the curve. The `mark` argument will mark particular thresholds along each models ROC plot. Notice how evenly spaced thresholds do not end up equal distances apart along a ROC plot (Figure 7):

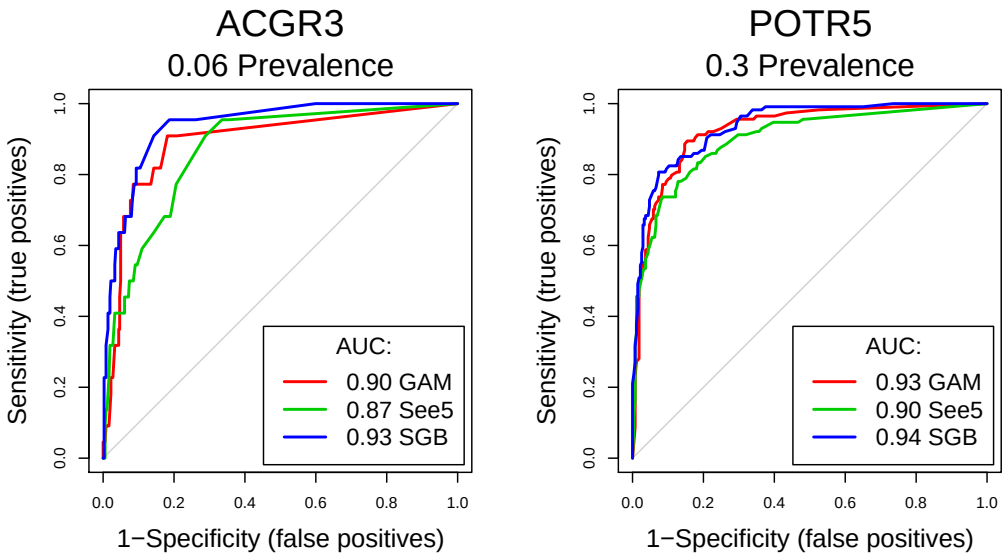

Figure 6: ROC plot for two species with similar model quality but differing prevalence.

```
R> par(oma = c(0, 0, 0, 0), mfrow = c(1, 2), cex = 0.7, cex.lab = 1.5)
R> sp <- c("ACGR3", "POTR5")
R> for (i in 1:2) {
+     DATA <- SPDATA[SPDATA$SPECIES == sp[i], ]
+     auc.roc.plot(DATA, color = TRUE, legend.cex = 1.4, mark = 5,
+         which.model = 3, main = "")
+     mtext(sp[i], side = 3, line = 2.5, cex = 1.6)
+     mtext(paste(Obs.prev[names(Obs.prev) == sp[i]], "Prevalence"),
+         side = 3, line = 0.5, cex = 1.3)
+ }
```

ROC plots make model quality visually dramatic. Here, three species with the same prevalence have very different model quality(Figure 8):

```
R> par(oma = c(0, 0, 2, 0), mar = c(4, 4, 4, 1), mfrow = c(1, 3),
+     cex = 0.7, cex.lab = 1.4, mgp = c(2, 0.5, 0))
R> sp <- c("JUSC2", "ABCO", "PICO")
R> for (i in 1:3) {
+     DATA <- SPDATA[SPDATA$SPECIES == sp[i], ]
+     auc.roc.plot(DATA, color = TRUE, legend.cex = 1.2, main = "")
+     mtext(sp[i], side = 3, line = 2, cex = 1.6)
+     mtext(paste(Obs.prev[names(Obs.prev) == sp[i]], "Prevalence"),
+         side = 3, line = 0.5, cex = 1.3)
+ }
```

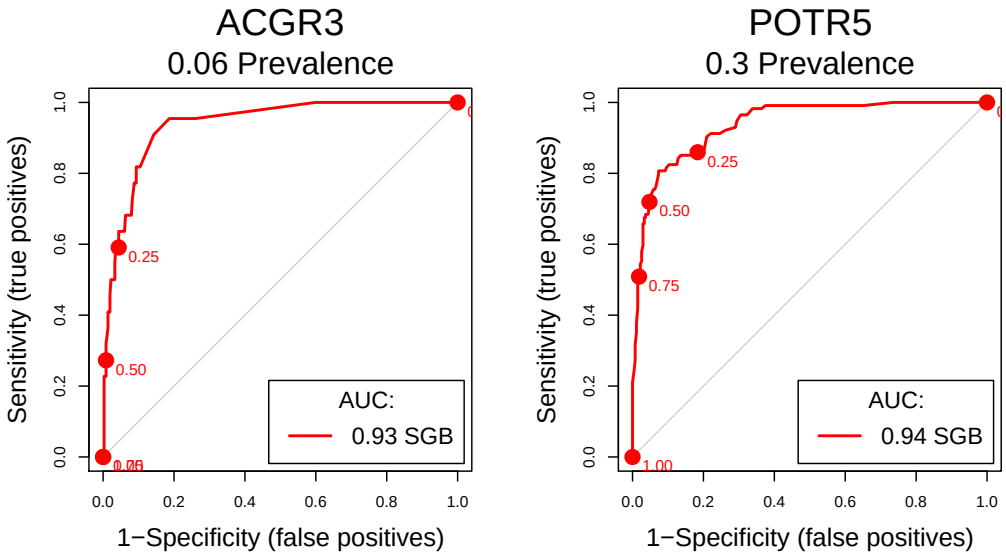

Figure 7: ROC plot for two species with similar model quality but differing prevalence, with evenly spaced thresholds marked along the ROC curves.

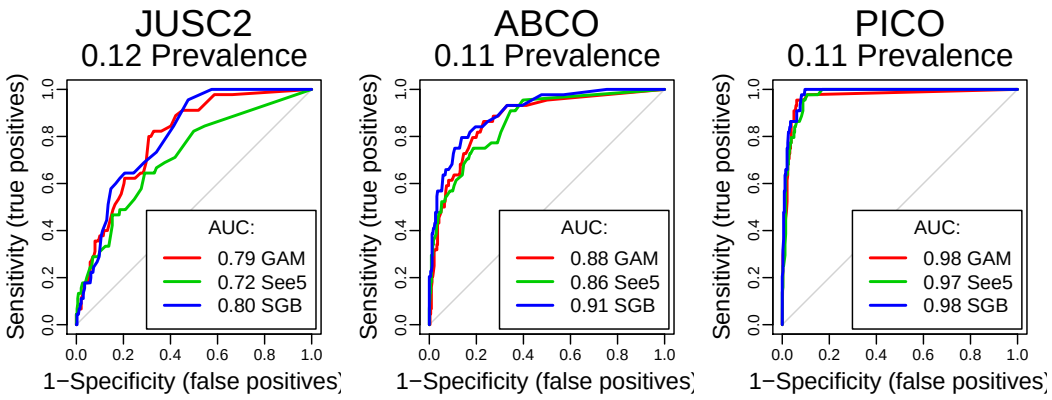

Figure 8: ROC plot for three species with similar prevalence but increasing model quality.

7. Optimizing thresholds

7.1. Calculating the optimal thresholds

It is possible to get rough ideas of good threshold choices by looking at the error graphs, and the accuracy and prevalence tables, however the **PresenceAbsence** library also includes the function `optimal.thresholds` to calculate tables of optimized thresholds, by a choice of 12 different criteria (Table 2).

By default, `optimal.thresholds` checks 101 thresholds evenly spaced between zero and one, and thus finds the optimal threshold to 2 significant digits. Some of the optimization criteria (`Sens=Spec`, `MaxSens+Spec`, `MaxKappa`, `MaxPCC`, `PredPrev=Obs`, `MinROCdist`, `Cost`) can result in tied threshold values. In these cases the ties are averaged to find the optimal threshold. With criteria that are optimized by finding minimum or maximum values, the `smoothing` argument can also be employed to deliberately average the best thresholds. Set `smoothing=10`, for example, to find the average of the ten best thresholds for a given criterion.

If the sample size is large, the precision can be increased by setting the argument `threshold` to larger numbers. Instead of the default `threshold=101`, set it to `threshold=1001`, or `threshold=10001`. Note that the precision will still be limited by the sample size, as the upper limit on precision is set by the number of unique thresholds in the data set.

To find the theoretically 'best' thresholds, would require calculating every possible unique threshold (not necessarily evenly spaced). This is not included in the package, but if these were calculated, they could be fed into the function `optimal.thresholds` via the `threshold` argument.

criterion ID number	criterion name	threshold description
1	Default	threshold set to 0.5
2	Sens=Spec	threshold where sensitivity equals specificity
3	MaxSens+Spec	threshold that maximizes sum of sensitivity and specificity
4	MaxKappa	threshold that maximizes Kappa
5	MaxPCC	threshold that maximizes PCC
6	PredPrev=Obs	threshold where predicted prevalence equals observed prevalence
7	ObsPrev	threshold set to Observed prevalence
8	MeanProb	threshold set to mean predicted probability
9	MinROCdist	threshold where ROC curve makes closest approach to (0,1)
10	ReqSens	highest threshold where sensitivity meets user defined requirement
11	ReqSpec	lowest threshold where specificity meets user defined requirement
12	Cost	threshold based on user-defined relative costs ratio

Table 2: Possible optimization criteria

The twelve possible criteria for optimizing thresholds are summarized in (Table 2). The `opt.methods` argument is specified by a vector of criteria from this list. The methods can be given by number, for example `opt.methods=1:10` or `opt.methods=c(1,2,4)`. They can also be given by name, for example `opt.methods = c("Default", "Sens=Spec", "MaxKappa")`.

The 12 criteria in more detail:

`Default` — The default criterion of setting `threshold = 0.5`.

`Sens=Spec` — The second criterion for optimizing threshold choice is by finding the threshold where sensitivity equals specificity. In other words, find the threshold where positive observations are just as likely to be wrong as negative observations.

`MaxSens+Spec` — The third criterion chooses the threshold that maximizes the sum of sensitivity and specificity. In other words, it is minimizing the mean of the error rate for positive observations and the error rate for negative observations. This is equivalent to maximizing $(sensitivity + specificity - 1))$, otherwise know as the Youden's index, or the True Skill Statistic. Note that while Youden's Index is independent of prevalence, Manel *et al.* (2001) found that using the sum of sensitivity and specificity to select a threshold does have an effect on the predicted prevalence, causing rare species to be given much lower thresholds than widespread species, as a result, the distribution of rare species can be over predicted.

`MaxKappa` — The forth criterion for optimizing the threshold choice finds the threshold that gives the maximum value of Kappa. Kappa makes full use of the information in the confusion matrix to asses the improvement over chance prediction.

`MaxPCC` — The fifth criterion is to maximize the total accuracy PCC. A warning: while it may seem like maximizing total accuracy would be the obvious goal, there are many problems with using PCC to assess model accuracy. For example, with species with very low prevalence, it is possible to maximize PCC simply by declaring the species absent at all locations. This is not a very useful prediction.

`PredPrev=Obs` — The sixth criterion is to find the threshold where the predicted prevalence is equal to the observed prevalence. This is a useful method when preserving prevalence is of prime importance. If you have observed prevalence data from another source, you can use the argument `obs.prev` to set the observed prevalence. Otherwise, `obs.prev` defaults to the observed prevalence from `DATA`.

`ObsPrev` — The seventh criterion is an even simpler variation, where you simply set the threshold to the Observed prevalence. It is nearly as good at preserving prevalence as method 6 and requires no computation. Again, if you have observed prevalence data from another source, you can use the argument `obs.prev` to set the observed prevalence. Otherwise, `obs.prev` defaults to the observed prevalence from `DATA`.

`MeanProb` — The eighth criterion sets the threshold to the mean probability of occurrence from the model results.

`MinROCdist` — The ninth criterion is to find the threshold that minimizes the distance between the ROC plot and the upper left corner of the unit square.

`ReqSens` — The tenth criterion allows the user to set a required sensitivity, and then finds the highest threshold (i.e. with the highest possible specificity) that still meets the required sensitivity. The user can decide that the model must miss no more than, for example 15% of the plots where the species is observed to be present. Therefore they require a sensitivity of at least 0.85. This will then find the threshold that has the highest possible specificity while

still meeting the required sensitivity. This may be useful if, for example, the goal is to define a management area for a rare species, and one wants to be certain that the management area does not leave populations unprotected.

`ReqSpec` — The eleventh criterion allows the user to set a required specificity, and then finds the lowest threshold (i.e. with the highest possible sensitivity) that still meets the required specificity. The user can decide that the model must miss no more than, for example 15% of the plots where the species is observed to be absent. Therefore they require a specificity of at least 0.85. This will then find the threshold that has the highest possible sensitivity while still meeting the required specificity. This may be useful if, for example, the goal is to determine if a species is threatened, and one wants to be certain not to over inflate the population by over declaring true absences as predicted presences.

Note: For the `ReqSens` and `ReqSpec` criteria, the required sensitivity (or specificity) can be changed to meet specific user requirements by the use of the `req.sens` and `req.spec` arguments. If unspecified, the defaults are `req.sens = 0.85` and `req.spec = 0.85`. Be aware, though, that if the model is poor, and the requirement is too strict, it is possible that the only way to meet it will be by declaring every single plot to be present (for `ReqSens`) or absent (for `ReqSpec`), which is not a very useful method of prediction.

`Cost` — The twelfth criterion balances the relative costs of false positive predictions and false negative predictions, as described by Fielding and Bell (1997). A slope is calculated as $(FPC/FNC)\,((1 - prevalence)\,/prevalence)$, where FPC is false positive costs and FNC is false negative costs. To determine the threshold, a line of this slope is moved from the top left of the ROC plot, till it first touches the ROC curve. If the arguments for false positive costs `FPC` and `FNC` are missing, the function will assume costs are equal.

Once again, if you have observed prevalence data from another source, you can use the argument `obs.prev` to set the observed prevalence. Otherwise, `obs.prev` defaults to the observed prevalence from `DATA`.

The criterion `Cost` can also be used for C/B ratio analysis of diagnostic tests. In this case `FPC` is equivalent to C (the net costs of treating non-diseased individuals) and `FNC` is equivalent to B (the net benefits of treating diseased individuals). For further information on the `Cost` criterion see Wilson *et al.* (2004) and Cantor *et al.* (1999).

Below is an example of using `optimal.thresholds` to calculate a threshold table. The criteria `ReqSens`, `ReqSpec` and `Cost` all depend on user defined requirements. In the first example these arguments are missing, and the function generated warning messages:

```
R> sp <- "QUGA"
R> DATA <- SPDATA[SPDATA$SPECIES == sp, ]
R> print(paste("Species:", sp))

[1] "Species: QUGA"

R> optimal.thresholds(DATA, opt.methods = 1:12)

         Method       GAM       See5       SGB
1       Default 0.5000000 0.5000000 0.5000000
2     Sens=Spec 0.1200000 0.1700000 0.1100000
```

```
3  MaxSens+Spec 0.3000000 0.0900000 0.1750000
4      MaxKappa 0.3000000 0.3900000 0.3000000
5        MaxPCC 0.4250000 0.6800000 0.4475000
6  PredPrev=Obs 0.4400000 0.3800000 0.2650000
7       ObsPrev 0.1476684 0.1476684 0.1476684
8      MeanProb 0.1484497 0.1482098 0.1344168
9    MinROCdist 0.3000000 0.1800000 0.1100000
10      ReqSens 0.1300000 0.0800000 0.0700000
11      ReqSpec 0.0900000 0.2700000 0.1400000
12         Cost 0.4250000 0.6800000 0.4700000
```

In this second example, the arguments `req.sens`, `req.spec`, `FPC` and `FNC` are used to supply these values:

```
R> sp <- "QUGA"
R> DATA <- SPDATA[SPDATA$SPECIES == sp, ]
R> print(paste("Species:", sp))

[1] "Species: QUGA"

R> optimal.thresholds(DATA, opt.methods = 1:12, req.sens = 0.9,
+     req.spec = 0.9, FPC = 2, FNC = 1)

         Method       GAM      See5       SGB
1       Default 0.5000000 0.5000000 0.5000000
2     Sens=Spec 0.1200000 0.1700000 0.1100000
3  MaxSens+Spec 0.3000000 0.0900000 0.1750000
4      MaxKappa 0.3000000 0.3900000 0.3000000
5        MaxPCC 0.4250000 0.6800000 0.4475000
6  PredPrev=Obs 0.4400000 0.3800000 0.2650000
7       ObsPrev 0.1476684 0.1476684 0.1476684
8      MeanProb 0.1484497 0.1482098 0.1344168
9    MinROCdist 0.3000000 0.1800000 0.1100000
10      ReqSens 0.0500000 0.0600000 0.0600000
11      ReqSpec 0.1900000 0.3600000 0.2200000
12         Cost 0.7250000 0.6800000 0.5700000
```

7.2. Adding optimal thresholds to graphs

The optimized thresholds can be added to the histogram plots, the error statistic plots, and to the ROC plots. To add optimal thresholds set the argument `opt.thresholds=TRUE`. (Note: If the argument `opt.methods` is supplied, `opt.thresholds` will default to `TRUE`, otherwise `opt.thresholds` defaults to `FALSE`.)

When the argument `add.opt.legend = TRUE` a legend is added giving the symbols used for each threshold criterion, as well as the value of each optimized threshold. In this case, the legend is giving the optimized thresholds for the SGB model for PICO.

Histogram plots

We will begin with a histogram plot for three species with similar prevalence, but increasing model quality (Figure 9). Notice how in the case of good models (such as the See5 model for PICO), the all optimized thresholds fall into the trough in the center of the double humped histogram. Poor models (such as the See5 model for JUSC2) do not have this double humped histogram, and so there is no single threshold that will neatly divide the data into present and absent groups. In these cases, the different optimization criteria tend to be widely separated, each representing a different compromise between the error statistics.

```
R> par(oma = c(0, 0, 3, 0), mar = c(4, 4, 4, 1), mfrow = c(1, 3), cex = 1)
R> sp <- c("JUSC2", "ABCO", "PICO")
R> mod <- 2
R> for (i in 1:3) {
+     presence.absence.hist(DATA = SPDATA[SPDATA$SPECIES ==
+         sp[i], ], which.model = mod, N.bars = 15, truncate.tallest = TRUE,
+         legend.cex = 0.6, opt.legend.cex = 0.6, opt.thresholds = TRUE,
+         opt.methods = c(1, 2, 10, 4), req.sens = 0.85, main = sp[i])
+     mtext(paste(Obs.prev[names(Obs.prev) == sp[i]], "Prevalence"),
+         side = 3, line = 0.5, cex = 1.1)
+ }

[1] "height of tallest bar truncated to fit on plot"
[1] "height of tallest bar truncated to fit on plot"
[1] "height of tallest bar truncated to fit on plot"

R> mtext(model.names[mod], side = 3, line = 0, cex = 1.5, outer = TRUE)
```

Error statistics as a function of threshold

Plots created with `error.threshold.plot` always include sensitivity and specificity. Some values of the argument `opt.methods` will result in adding additional accuracy statistics to the graphs. These lines for the error statistics specified in `opt.methods` will be plotted, even if `opt.thresholds=FALSE`. Four criteria from `opt.methods` add lines to `error.threshold.plot`: `MaxKappa`, `MaxPCC`, `MinROCdist`, and `MaxSens+Spec`. The criteria `MaxKappa` and `MaxPCC` add Kappa and PCC, respectively. The criterion `MinROCdist` adds the distance from the ROC curve to the upper left corner of the ROC plot. Finally, the criterion `MaxSens+Spec` adds a line showing $(Sensitivity + Specificity - 1)$. In addition to these lines representing the error statistics, for `opt.methods` criteria of `ReqSens` and `ReqSpec`, a light gray horizontal line is drawn to mark the required sensitivity or specificity.

The default criteria are `opt.methods=1:4` which results in lines for sensitivity, specificity and Kappa. This is why the Kappa line was drawn in the error verses threshold plots in the previous section, even though `opt.thresholds` had defaulted to `FALSE`.

Here are the error plots for 2 species (JUSC2 and PICO) with optimized thresholds added to the plots (Figure 10). Both species have the same prevalence, but PICO had better model quality than JUSC2 for all three types of models. When examining the graphs for the higher

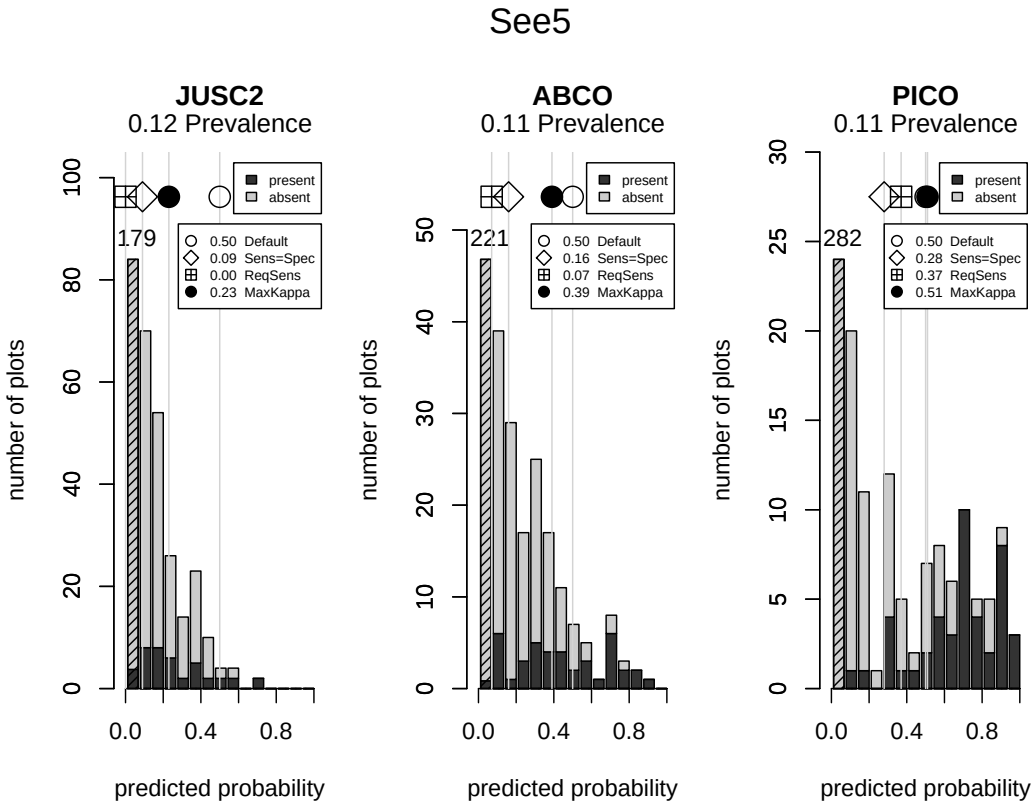

Figure 9: Histogram plot for three species with similar prevalence but increasing model quality, with thresholds optimized by four criteria.

quality models, notice how sensitivity and specificity cross at a higher value, and also, the error statistics show good values across a broader range of thresholds.

Note that while the symbols in the optimal threshold legend are correct for all 6 plots, the actual thresholds given in the legend are specific to that particular plot (in this case, species PICO, model SGB). In this particular case, if both legends had been included on all 6 plots, there would not be room to see the plots themselves.

```
R> par(oma = c(0, 5, 0, 0), mar = c(3, 3, 3, 1), mfrow = c(2,
+     3), cex = 0.7, cex.lab = 1.4, mgp = c(2, 0.5, 0))
R> sp <- c("JUSC2", "PICO")
R> for (i in 1:2) {
+     DATA = SPDATA[SPDATA$SPECIES == sp[i], ]
+     for (mod in 1:3) {
+         if (i == 2 & mod == 2) {
+             legend.flag <- TRUE
+         }
+         else {
+             legend.flag <- FALSE
```

```
+         }
+         if (i == 2 & mod == 3) {
+             opt.legend.flag <- TRUE
+         }
+         else {
+             opt.legend.flag <- FALSE
+         }
+         error.threshold.plot(DATA, which.model = mod, color = TRUE,
+             opt.thresholds = TRUE, opt.methods = c(1, 2, 10,
+                 4), req.sens = 0.9, add.legend = legend.flag,
+             add.opt.legend = opt.legend.flag, legend.cex = 1.1,
+             opt.legend.cex = 1.1, vert.lines = FALSE)
+         if (mod == 1) {
+             mtext(sp[i], side = 2, line = 6.3, cex = 1.6)
+             mtext(paste("(", Obs.prev[names(Obs.prev) == sp[i]],
+                 "Prevalence )"), side = 2, line = 4.3, cex = 1.3)
+         }
+     }
+ }
```

The argument `vert.lines` can be used to specify how the optimized thresholds are marked on the plot: `vert.lines=TRUE` a vertical line is drawn at each optimized threshold, and labeled along the top of the plot; `vert.lines=FALSE` the thresholds are marked along the error statistics (the default).

The next graph shows three species with increasing prevalence, and illustrates how certain optimization criteria (in this case setting the threshold so that sensitivity equals specificity) are dependent on prevalence (Figure 11). This graph also illustrates how the different threshold criteria tend to converge as prevalence increases. Finally, the graph for the low prevalence species (ACGR3) drives home why maximizing PCC does not work well for species with very low or very high prevalence. In the case of ACGR3, maximizing PCC gives a threshold of 0.95, which results in an extremely low sensitivity, therefore the model predictions will miss nearly all of the true presences.

```
R> par(oma = c(0, 0, 2, 0), mar = c(4, 4, 4, 1), mfrow = c(1, 3),
+     cex = 0.7, cex.lab = 1.4, mgp = c(2, 0.5, 0))
R> sp <- c("ACGR3", "QUGA", "JUOS")
R> mod = 1
R> for (i in 1:3) {
+     DATA = SPDATA[SPDATA$SPECIES == sp[i], ]
+     if (i == 2) {
+         legend.flag <- TRUE
+     }
+     else {
+         legend.flag <- FALSE
+     }
+     if (i == 3) {
+         opt.legend.flag <- TRUE
```

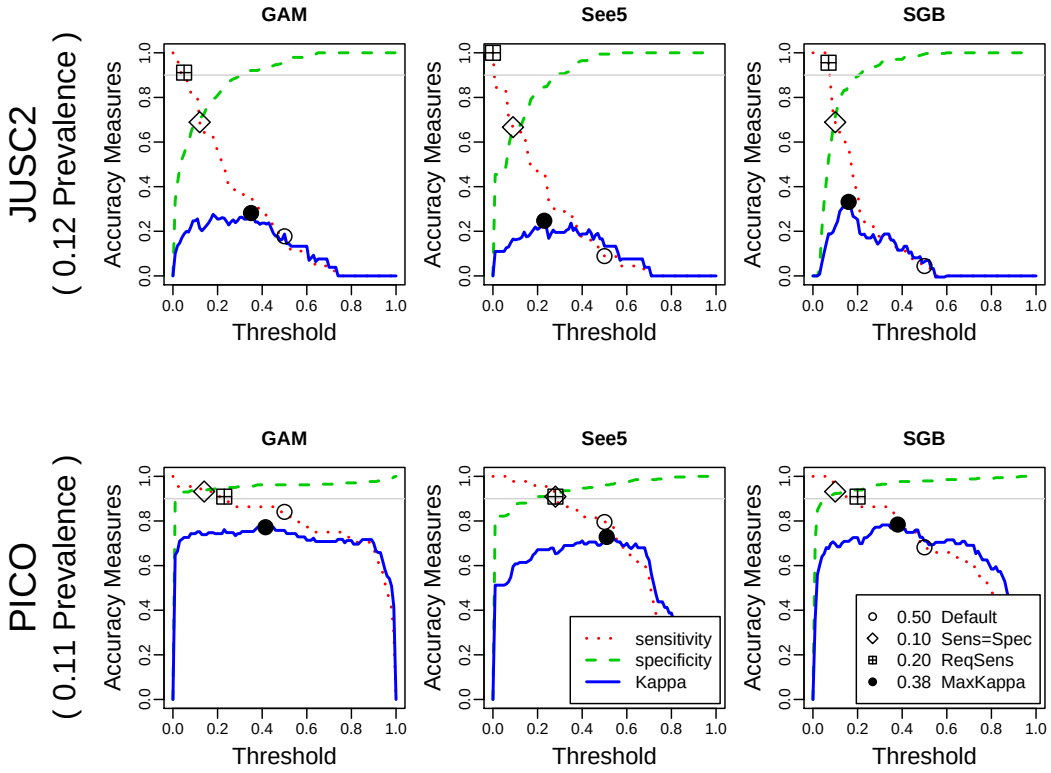

Figure 10: Error as a function of threshold for two species with similar prevalence, but differing model quality, with optimized thresholds marked along plots

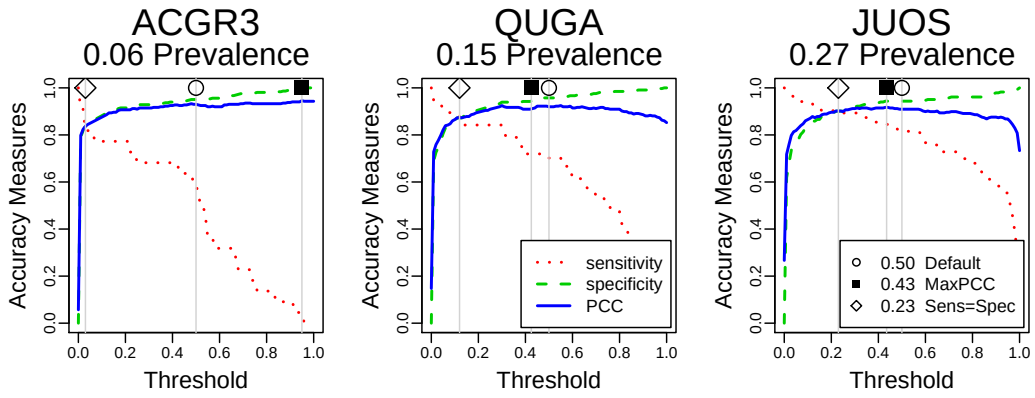

Figure 11: Error as a function of threshold for three species similar model quality but with increasing prevalence, with optimized thresholds marked on plots.

```
+     }
+     else {
+         opt.legend.flag <- FALSE
+     }
+     error.threshold.plot(DATA, which.model = mod, color = TRUE,
+         opt.thresholds = TRUE, opt.methods = c("Default",
+             "MaxPCC", "Sens=Spec"), add.legend = legend.flag,
+         add.opt.legend = opt.legend.flag, legend.cex = 1.1,
+         opt.legend.cex = 1.1, vert.lines = TRUE, main = "")
+     mtext(sp[i], side = 3, line = 2, cex = 1.6)
+     mtext(paste(Obs.prev[names(Obs.prev) == sp[i]], "Prevalence"),
+         side = 3, line = 0.5, cex = 1.3)
+ }
```

ROC plots

As with previous plotting functions, the optimized thresholds can be added to the ROC plots. This can get hard to read if all the models are on the same plot, so here the models are split onto separate plots (Figure 12):

```
R> par(oma = c(0, 5, 0, 0), mar = c(3, 3, 3, 1), mfrow = c(2,
+     3), cex = 0.7, cex.lab = 1.4, mgp = c(2, 0.5, 0))
R> sp <- c("JUSC2", "PICO")
R> for (i in 1:2) {
+     DATA <- SPDATA[SPDATA$SPECIES == sp[i], ]
+     for (mod in 1:3) {
+         auc.roc.plot(DATA, color = mod + 1, which.model = mod,
+             main = model.names[mod], opt.thresholds = TRUE,
+             opt.methods = c(1, 4, 6), mark.numbers = TRUE,
+             mark.color = FALSE, legend.cex = 1.1, opt.legend.cex = 1.1)
+         if (mod == 1) {
+             mtext(sp[i], side = 2, line = 6.3, cex = 1.6)
+             mtext(paste("(", Obs.prev[names(Obs.prev) == sp[i]],
+                 "Prevalence )"), side = 2, line = 4.3, cex = 1.3)
+         }
+     }
+ }
```

8. Calibration plots

Calibration plots provide a goodness-of-fit plot for presence-absence models, as described by Pearce and Ferrier (2000), Vaughan and Ormerod (2005), and Reineking and Schröder (2006). Sensitivity, specificity, Kappa, and AUC all assess a model's discriminatory ability, that is, how accurately can the model divide the locations into presences and absences. Model calibration, on the other hand, concerns the accuracy of the predicted probabilities, for example, do 40% of the locations assigned a probability of 0.4 actually have the species present.

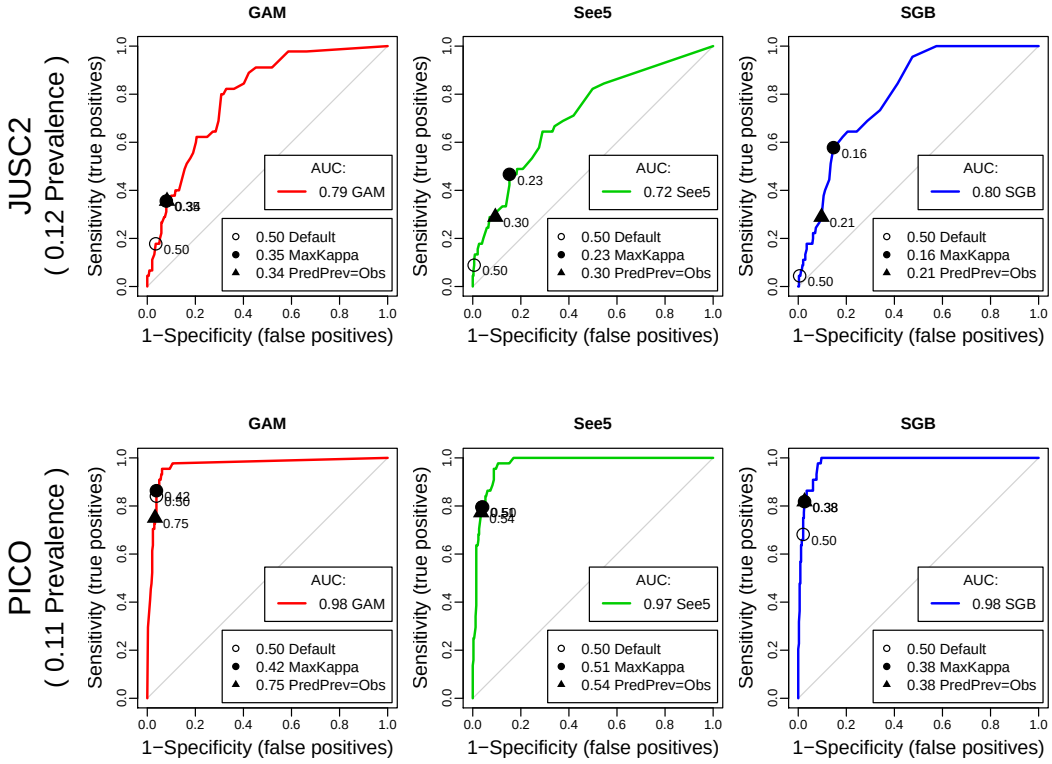

Figure 12: ROC plot for two species with similar prevalence but different model quality, with optimal thresholds marked along ROC curve.

This type of plot must be evaluated in light of the intended purpose of the model. Unlike a typical goodness-of-fit plot from a linear regression model, with presence-absence data having all the points lay along the diagonal does not necessarily imply an ideal model. The best bin plot for presence-absence data depends on how the model is to be used.

If the model is to be used to produce probability maps, then it is indeed desirable that the predicted probability is well calibrated to the true probability. For example, 80% of plots with predicted probability of 0.8 should actually have observed presence. In this case, having all the bins along the diagonal does indicate a good model.

However, if model is to be used simply to discriminate between species presence and absence, then all that is required is that some threshold exists (not necessarily 0.5) where every plot with a lower predicted probability is observed absent, and every plot with a higher predicted probability is observed present. In this case, a good model will not necessarily (in fact, will rarely) have all the bins along the diagonal. (Note: to check a models discriminatory ability purpose `presence.absence.hist` will produce more useful diagnostics.)

Calibration plots can also provide a useful check on the selection of training and test datasets. If all the bins lie above the diagonal, or all the bins lie below the diagonal, it may indicate that the training and test datasets have different prevalence. In this case, it may be worthwhile to re-examine the initial data selection.

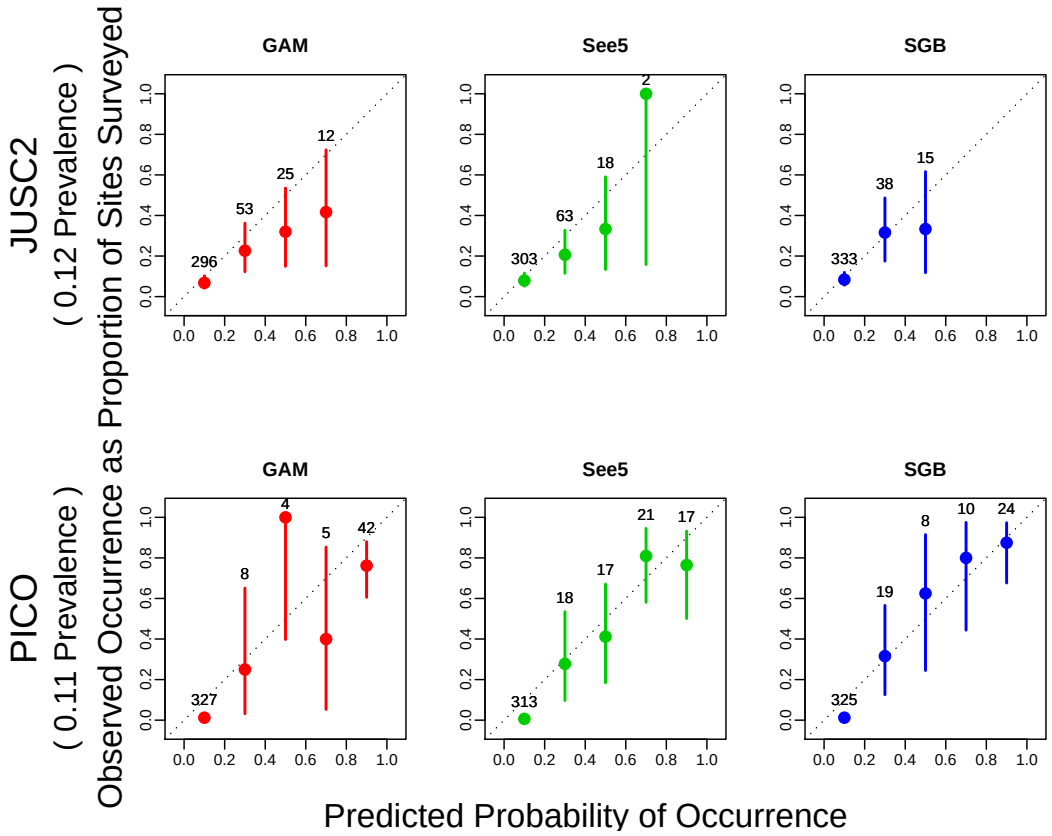

Figure 13: Calibration plot for two species with similar prevalence but different model quality.

In the calibration plots provided in the **PresenceAbsence** package, the locations are grouped into bins based on their predicted values, and then the ratio of plots with observed values of present verses the total number of plots in that bin is calculated. The confidence interval for each bin is also included, and the total number of plots is labeled above each the bin. Here, we produce calibration plots for JUSC2 and PICO (Figure 13):

```
R> par(oma = c(0, 5, 0, 0), mar = c(3, 3, 3, 1), mfrow = c(2,
+     3), cex = 0.7, cex.lab = 1.4, mgp = c(2, 0.5, 0))
R> sp <- c("JUSC2", "PICO")
R> for (i in 1:2) {
+     DATA = SPDATA[SPDATA$SPECIES == sp[i], ]
+     for (mod in 1:3) {
+         calibration.plot(DATA, which.model = mod, color = mod +
+             1, main = model.names[mod], xlab = "", ylab = "")
+         if (mod == 1) {
+             mtext(sp[i], side = 2, line = 6.3, cex = 1.6)
+             mtext(paste("(", Obs.prev[names(Obs.prev) == sp[i]],
+                 "Prevalence )"), side = 2, line = 4.3, cex = 1.3)
+         }
```

```
+     }
+ }
R> mtext("Predicted Probability of Occurrence", side = 1, line = -1,
+     cex = 1.4, outer = TRUE)
R> mtext("Observed Occurrence as Proportion of Sites Surveyed",
+     side = 2, line = -1, cex = 1.4, outer = TRUE)
```

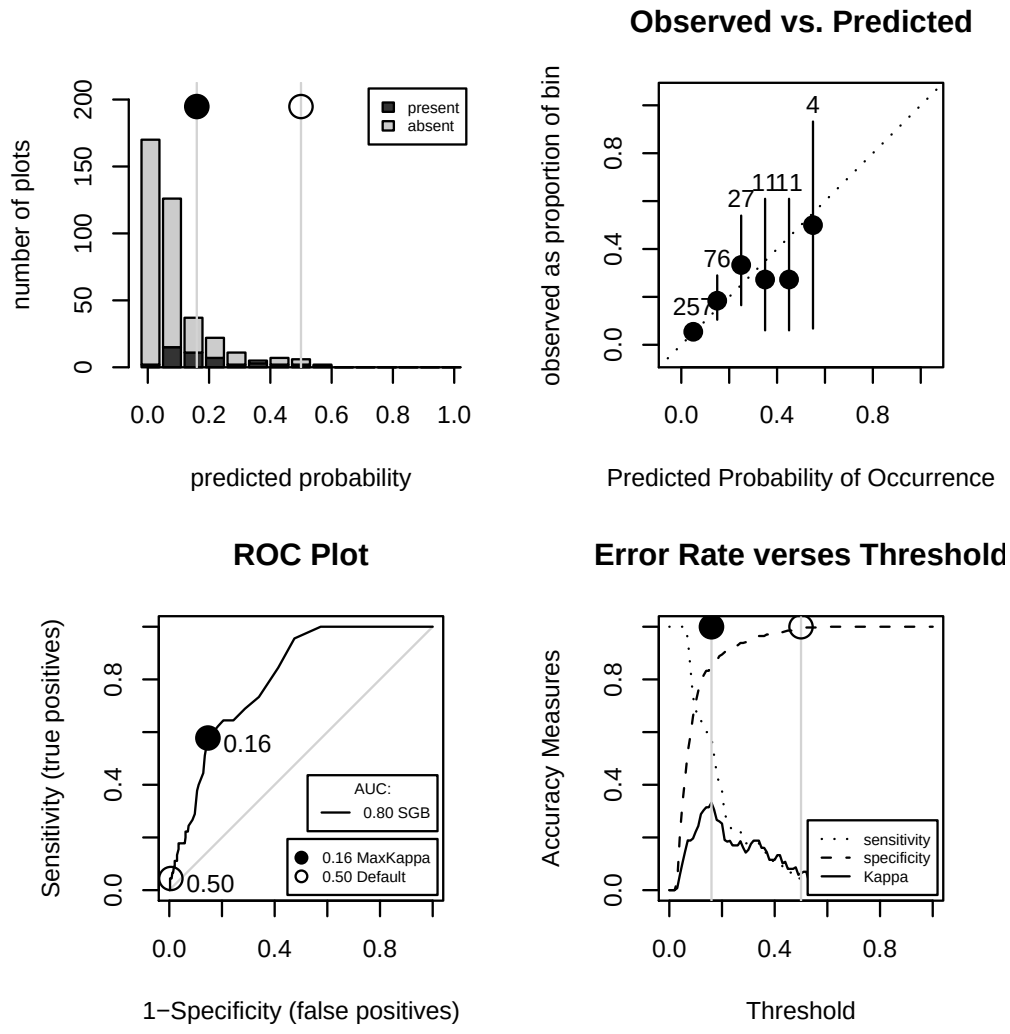

Figure 14: Summary plot for species JUSC2.

9. Summary plots

Finally, the function `presence.absence.summary` produces a useful summary page of the four types of presence-absence plots, along with optimal thresholds and AUC value (Figure 14):

```
R> par(mar = c(4, 4, 4, 1))
R> sp <- "JUSC2"
R> DATA = SPDATA[SPDATA$SPECIES == sp, ]
R> MAIN <- paste(sp, "(", Obs.prev[names(Obs.prev) == sp], "Prevalence ) - ",
+     model.names[mod])
R> presence.absence.summary(DATA, opt.methods = c("MaxKappa",
+     "Default"), which.model = mod, N.bins = 10, N.bars = 15,
+     main = MAIN, legend.cex = 0.6, opt.legend.cex = 0.6, vert.lines = TRUE)
```

10. Conclusion

We have demonstrated the use of the **PresenceAbsence** package in a typical species mapping scenario. We have explored the model output both graphically and analytically, and examined the effects of optimizing the cut-off threshold by several different criteria. This analysis and decision making process would be similar in other binary classification datasets. We hope we have shown that the **PresenceAbsence** package provides a useful toolkit for map makers and map users when evaluating the probability surface output from a presence-absence model, and for translating the probability surface into species distribution maps for use in the field.

References

Anderson R, Lew D, Peterson A (2003). "Evaluating Predictive Models of Species' Distributions: Criteria for Selecting Optimal Models." *Ecological Modelling*, **162**, 211–232.

Atkinson B, Mahoney D (2004). ***roc**: S-PLUS ROC Functions*. Mayo Clinic. URL http://mayoresearch.mayo.edu/mayo/research/biostat/splusfunctions.cfm.

Cantor S, Sun C, Tortolero-Luna G, Richards-Kortum R, Follen M (1999). "A Comparison of C/B Ratios from Studies Using Receiver Operating Characteristic Curve Analysis." *Journal of Clinical Epidemiology*, **52(9)**, 885–892.

Carey V, Redestig H (2007). ***ROC**: Utilities for ROC, with Uarray Focus*. R package version 1.12.0, URL http://www.bioconductor.org/.

Carstensen B, Plummer M, Läärä E, Myatt M, Clayton D, et al (2007). ***Epi**: A Package for Statistical Analysis in Epidemiology*. R package version 0.7.0, URL http://CRAN.R-project.org/.

Caruana R (2004). ***PERF**: Performance Evaluation Code*. ACM Special Interest Group on Knowledge Discovery and Data Mining. URL http://www.sigkdd.org/kddcup/index.php?section=2004&method=soft.

Congalton R (1991). "A Review of Assessing the Accuracy of Classifications of Remotely Sensed Data." *Remote Sensing of Environment*, **37(1)**, 35–46.

DeLong E, Delong D, Clarke-Pearson D (1988). "Comparing Areas Under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach." *Biometrics*, **44(3)**, 387–394.

Eng J (2006). ***jrocfit.org**: Web-based Calculator for ROC Curves*. Johns Hopkins University. URL http://www.jrocfit.org/.

Farrand J, Langford T (2002). ***ROCOn**: A Tool for Visualising ROC Graphs*. University of Bristol. URL http://www.cs.bris.ac.uk/Research/MachineLearning/rocon/.

Fielding A, Bell J (1997). "A Review of Methods for the Assessment of Prediction Errors in Conservation Presence/Absence Models." *Environmental Conservation*, **24(1)**, 38–49.

Freeman E (2007). *PresenceAbsence: An R Package for Presence-Absence Model Evaluation*. USDA Forest Service, Rocky Mountain Research Station, 507 25th Street, Ogden, UT, USA. URL http://CRAN.R-project.org/.

Greiner M, Pfeiffer D, Smith R (2000). "Principles and Practical Application of the Receiver-Operation Characteristic Analysis for Diagnostic Tests." *Preventive Veterinary Medicine*, **45**, 23–41.

Guisan A, Hofer U (2003). "Predicting Reptile Distributions at the Mesoscale: Relation to Climate and Topography." *Journal of Biogeography*, **30**, 1233–1243.

Guisan A, Thuiller W (2005). "Predicting Species Distribution." *Ecology Letters*, **8**, 993–1009.

Hanley JA, McNeil BJ (1982). "The Meaning and Use of the Area Under a ROC Curve." *Radiology*, **143**, 29–36.

Hirzel A, Le Lay G, Helfer V, Randin C, Guisan A (2006). "Evaluating the Ability of Habitat Suitability Models to Predict Species Presences." *Ecological Modelling*, **199**, 142–152.

Insightful Corp (2003). *S-PLUS Version 6.2*. Seattle, WA. URL http://www.insightful.com/.

Manel S, Dias J, Ormerod S (1999). "Comparing Discriminant Analysis, Neural Networks and Logistic Regression for Predicting Species Distribution: A Case Study with a Himalayan River Bird." *Ecological Modelling*, **120**, 337–347.

Manel S, Williams H, Ormerod S (2001). "Evaluating Presence-Absence Models in Ecology: the Need to Account for Prevalence." *Journal of Applied Ecology*, **38**, 921–931.

Moisen G, Freeman E, Blackard J, Frescino T, Zimmerman N, Edwards T (2006). "Predicting Tree Species Presence and Basal Area in Utah: A Comparison of Stochastic Gradient Boosting, Generalized Additive Models, and Tree-Based Methods." *Ecological Modelling*, **199**, 176–187.

NCAR - Research Application Program (2007). ***verification**: Forecast Verification Utilities*. R package version 1.20, URL http://CRAN.R-project.org/.

Pearce J, Ferrier S (2000). "Evaluating the Predicting Performance of Habitat Models Developed Using Logistic Regression." *Ecological Modelling*, **133**, 225–245.

R Development Core Team (2007). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Reineking B, Schröder B (2003). "Computer-Intensive Methods in the Analysis of Species-Habitat Relationships." In B Breckling, H Reuter, A Mitwollen (eds.), "Gene, Bits und Ökosysteme - Implikationen neuer Technologien für die ökologische Theorie," pp. 165–182. Peter Lang, Frankfurt am Main.

Reineking B, Schröder B (2006). "Constrain to Perform: Regularization of Habitat Models." *Ecological Modelling*, **193**, 675–690.

Ridgeway G (2006). ***gbm**: Generalized Boosted Regression Models*. R package version 1.5-5, URL http://CRAN.R-project.org/.

Schröder B (2006). ***ROC_AUC**: Evaluating the Predictive Performance of Species Distribution Models*. URL http://brandenburg.geoecology.uni-potsdam.de/users/schroeder/download.html.

Sing T, Sander O, Beerenwinkel N, Lengauer T (2005). "**ROCR**: Visualizing Classifier Performance in R." *Bioinformatics*, **21**(20), 3940–3941.

Smyth G (2005). "**Limma**: Linear Models for Microarray Data." In R Gentleman, V Carey, S Dudoit, R Irizarry, W Huber (eds.), "Bioinformatics and Computational Biology Solutions using R and Bioconductor," pp. 397–420. Springer-Verlag, New York.

Swets JA (1988). "Measuring the Accuracy of Diagnostic Systems." *Science*, **240**, 1285–1293.

Tuszynski J (2007). ***caTools**: Moving Window Statistics, GIF, Base64, ROC AUC, etc.* R package version 1.8, URL http://CRAN.R-project.org/.

Vaughan I, Ormerod S (2005). "The Continuing Challenges of Testing Species Distribution Models." *Journal of Applied Ecology*, **42**, 720–730.

Wilson K, Westphal M, Possingham H, Elith J (2004). "Sensitivity of Conservation Planning to Different Approaches to Using Predicted Species Distribution Data." *Biological Conservation*, **22(1)**, 99–112.

Affiliation:

Elizabeth A. Freeman
US Forest Service
Rocky Mountain Research Station
507 25th Street
Ogden, UT 84401, United States of America
E-mail: eafreeman@fs.fed.us

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 23, Issue 11 *Submitted:* 2007-04-19
January 2008 *Accepted:* 2008-01-08