EEB 5301 **February 2017**

Species Distribution Models in R

R commands are shown in **red**. After typing any of these lines in the RUI gui, you must hit the return key. Output produced in the R Console is shown in **blue**. Comment lines are shown in **green**.

A. Getting ready for MaxEnt and Boosted Regression Trees

(1) Install the following packages: sp, dismo, maptools, rJava, randomForest. (Instructions for downloading and installing a package are given in "Matrices in PopBio.pdf", which also shows how to download manuals for packages.)

(2) Download MaxEnt from
https://www.cs.princeton.edu/~schapire/maxent/
It's free, but you are asked to give your name and e-mail address.

(3) One of the downloaded MaxEnt files is called **maxent.jar**. You need to put it into the 'java' folder of the 'dismo' package. First, be sure that you have installed the dismo package on your computer. Then, load the library and execute this command in R or RStudio:

```
system.file("java", package="dismo")
```

The result will show the path of the folder into which you must put maxent.jar. It will look something like this:

```
"C:/Users/Adams/Documents/R/win-library/3.3/dismo/java"
```

Your path will be different. Put maxent.jar in that folder.

(4) To reduce typing time and frustration, I have uploaded the R script for this tutorial to the course's HuskyCT site. It's called "**Species_Distribution_Models.R"**. Instead of reading this document, you can work through that file reading the comments.

(5) For the problem set, put the **fishers.csv** file (available on the HuskyCT site for the course) in your working folder. (You can make or choose a folder for this project on you computer. In R, set the working directory to that folder.)

(6) This tutorial is a much shortened version of one by Hijmans and Elith, on which I have depended heavily. Their longer tutorial can be downloaded from:

https://cran.r-project.org/web/packages/dismo/vignettes/sdm.pdf

Hijmans, R. J., and J. Elith. 2017. Species distribution modeling with R.

The Hijmans and Elith tutorial covers many more approaches to species distribution modeling and the steps need to clean and check a data set.

(7) You may need to download and install Java (or you may already have it installed on your computer). It’s free:

https://www.java.com/en/download/

Even if Java is already installed, when loading the rJava package or fitting a model in MaxEnt (below), you may get an error message if your version of Java doesn’t match your version of R (e.g., if you’re using the 32 bit version of Java and the 64 bit version of R).

```
Error : .onLoad failed in loadNamespace() for 'rJava', details:
call: fun(libname, pkgname)
error: JAVA_HOME cannot be determined from the Registry
```

If you get that error message, download and install the version of Java (32 bit or 64 bit) that matches your version of R.

B. Reading in the occurrence data and predictors

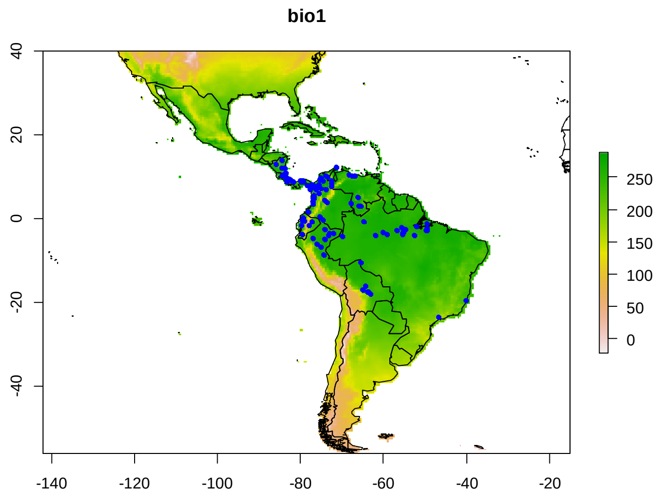
As an example, we can use a data set included with the dismo package on a species of sloth, *Bradypus variegatus*, a highly charismatic animal.

```
file <- paste(system.file(package="dismo"), "/ex/bradypus.csv", sep="")
bradypus <- read.table(file, header=TRUE, sep=',')
# Remove the first column, which isn't needed.
bradypus <- bradypus[,-1]
# wrld_simpl is from maptools and shows country outlines
data(wrld_simpl)
```

Read in the predictor layers, which are mostly climate data. hese are included with the dismo package and have been clipped to the world region that includes the range of *B. variegatus*,

```
files <- list.files(path=paste(system.file(package="dismo"),
                    '/ex', sep=''), pattern='grd', full.names=TRUE )
predictors <- stack(files)
# # Drop  the biomes layer, which is categorical rather than continuous
dropLayer(predictors, 'biome')
```

The SDM_R_tutorial.R file includes some commands that let you see the names of the predictors and plot them along with the *R. variegatus* locations.



C. Select background points; Set up training and test sets

This is a “presence-only” data set. Locations can be chosen randomly from the landscape. These are conceived of as background points or pseudo-absences depending on the modeling approach.

The following commands randomly divide the rows of the bradypus data set into five groups. Those in group 1 are held aside as a test set and the rest are use as the training set of presences. (Thus, the training set is four times as large as the test set.)

```
group <- kfold(bradypus, 5)
# The training set for presence sites consists of all rows for which
#   the randomly assigned group number is not equal to 1.
pres_train <- bradypus[group != 1, ]
# The test set for presence sites consists of rows with group number 1.
pres_test <- bradypus[group == 1, ]
```

Following the Hijmans and Elith tutorial, a restricted extent of the landscape is chosen, which speeds things up. The limits of the extent are

```
extb = extent(-90, -32, -33, 23)
```

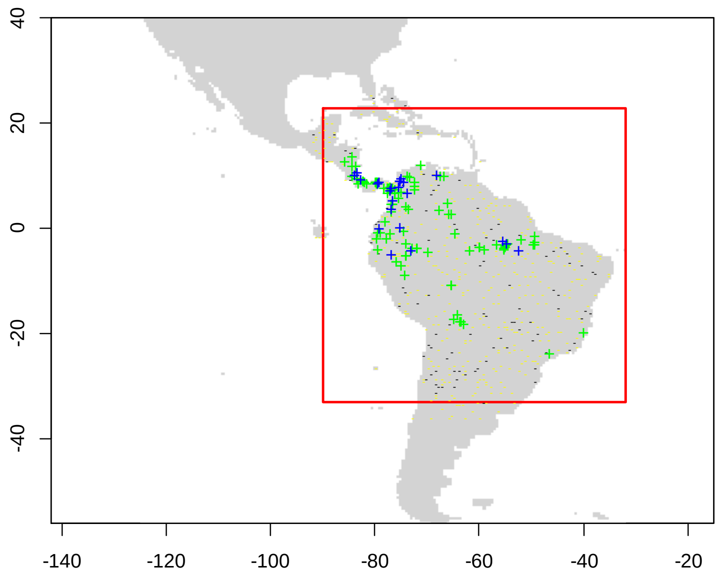
The four numbers are minimum longitude, maximum longitude, minimum latitude, and maximum latitude.

Now randomly select 500 background points from the landscape, within extent just defined. The columns are renamed from “x” and “y” to “lon” and “lat”. The 500 background points are also divided into a training and a test set.

```
backgr <- randomPoints(predictors, 500, ext=extb)
colnames(backgr) = c('lon', 'lat')  # renames the columns
```

```
# Select the training and test set for the background points
group <- kfold(backgr, 5)
backg_train <- backgr[group != 1, ]
backg_test <- backgr[group == 1, ]
```

The SDM_R_tutorial.R file shows how to plot the data on a map. The green plusses are the training set of presences, the blue plusses are the test set of presences, the yellow minuses are the training set of background points, and the black minuses are the test set of background points.



The two training sets are now combined into one data frame.

```
train <- rbind(pres_train, backg_train)
```

The next command produces a vector of 93 1's (for the training sites with bradypus) and 500 0's (for the background sites).

```
pb_train <- c(rep(1, nrow(pres_train)), rep(0, nrow(backg_train)))
```

D. Find the values of the predictors at each locality.

First, extract values of the predictors (bio1, bio5, etc.) at all the training sites and put them into a matrix called “envtrain.”

```
envtrain <- extract(predictors, train)
```

It is then turned into a data frame with the first column, “pa”, indicating presence (1) or absence (0). The presence and absence values are in “pb” which was created at the end of section C (above).

```
envtrain <- data.frame( cbind(pa=pb_train, envtrain) )
```

This command displays the first few rows:

```
head(envtrain)  # display the first few rows
```

```
  pa bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8
1  1  263  1639   724    62  338  191  147  261
2  1  263  1639   724    62  338  191  147  261
3  1  253  3624  1547   373  329  150  179  271
4  1  243  1693   775   186  318  150  168  264
5  1  243  1693   775   186  318  150  168  264
6  1  252  2501  1081   280  326  154  172  270
```

Now the values of the predictors are extracted for all of the test sites

```
testpres <- data.frame( extract(predictors, pres_test) )
testbackg <- data.frame( extract(predictors, backg_test) )
```

E. Use MaxEnt to fit a species distribution model.

First, run these two lines to see if the maxent.jar file is found in the expected folder. If not, you will get an error message, in which case go back to step A3 above. If the file is where it’s supposed to be, there will be no message.

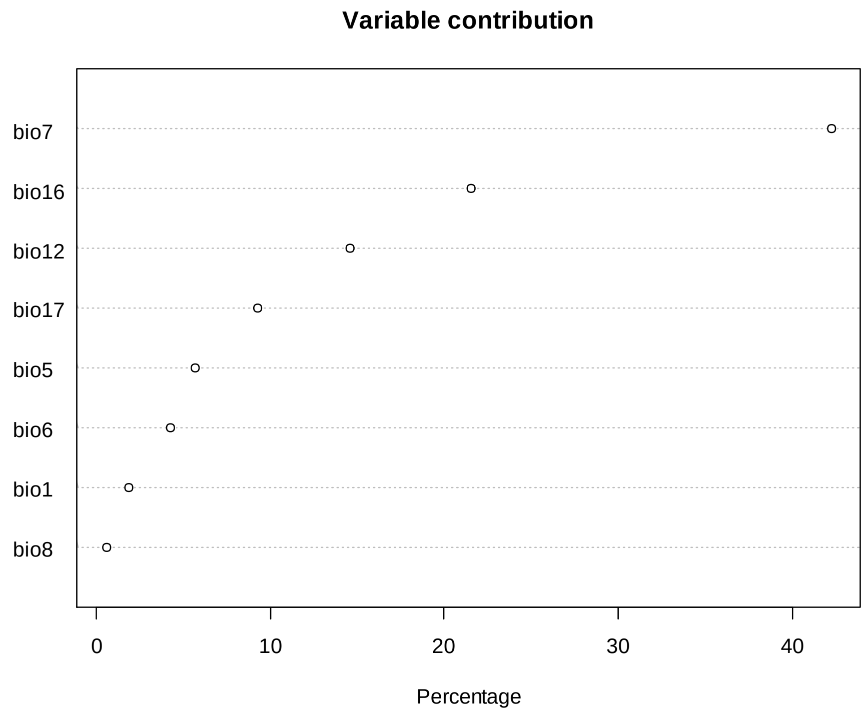
```
jar <- paste(system.file(package="dismo"), "/java/maxent.jar", sep='')
if (file.exists(jar)) {
  xm <- maxent(predictors, pres_train)
  plot(xm)
} else {
  cat('cannot run this example because maxent is not available')
  plot(1)
}
```

The command to build the model is easy, but it may take awhile to finish for larger data sets. The results are put into an object called “xm”.

```
xm <- maxent(predictors, pres_train)
```

The plot command shows the contribution of each variable.

```
plot(xm)
```



Response plots can be generated for all predictors

```
response(xm)
```

or just for a subset. Notice that bio7 and bio16 contribute the most. They are the 7th and 3rd predictors respectively. For just those two response plots:

```
response(xm, var=c(7,3))
```

Evaluate the fit of the model and display the results:

```
e <- evaluate(pres_test, backg_test, xm, predictors)
e
```

```
class          : ModelEvaluation
n presences    : 23
n absences     : 100
AUC            : 0.8221739
cor            : 0.4374661
max TPR+TNR at : 0.1587269
```

Finally, make a pair of plots. The first shows the predicted relative probabilities of occurrence over the landscape. Note that the “predict” command may take awhile to complete.

```
px <- predict(predictors, xm, ext=extb, progress='')
```

This command sets up the plotting area to show two plots side by side (1 row, 2 columns). When you want to go back to displaying only one plot, use par(mfrow=c(1,1)). But not yet.

```
par(mfrow=c(1,2))
```

The first plot is a map of the predicted values and overlay the country boundaries.

```
plot(px, main='Maxent, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
```

Choose a threshold value along the continuous scale of relative probabilities. This artificially divides the landscape into places where the species is expected to be present or absent. But that binary division depends on the threshold. (It’s a bit like dividing people into “tall” or “short” based on some threshold height.) The “spec_sens” option finds the threshold “at which the sum of the sensitivity (true positive rate) and specificity (true negative rate) is highest.”

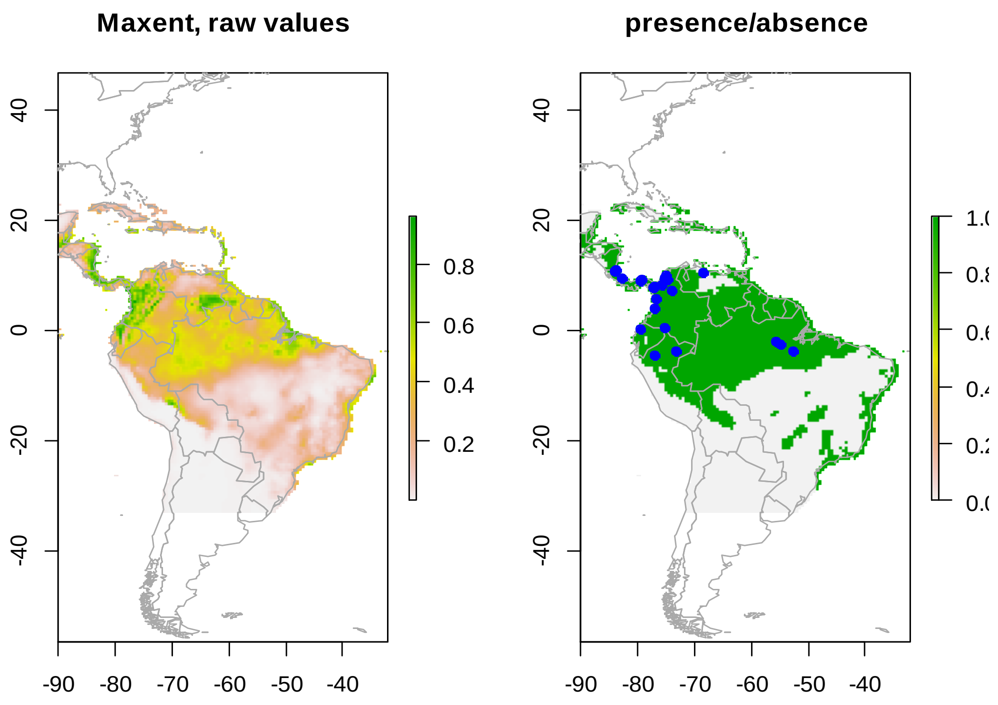
```
tr <- threshold(e, 'spec_sens')                    # choose a threshold value
```

The second plot shows the map divided into the two regions. The presence training sites are shown on the map.

```
plot(px > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```

Alternatively, you could plot the presence test sites.

```
points(pres_test, pch= 19, col='blue')
```



F. Use boosted regression trees to fit a species distribution model.

Regression trees can be used to model binary outcomes, in which case it is a classification model, or continuous outcomes, in which case it is a regression model. In this case, the outcome is binary (presence or absence), but Hijmans and Elith indicate that regression models nonetheless often provide better predictions. Here, I follow that suggestion. (If you want to try a classification model instead, see their tutorial.) Because we are fitting a regression to a binary outcome, you will see a warning message, which you can ignore:

```
The response has five or fewer unique values.  Are you sure you want to do
regression?
```

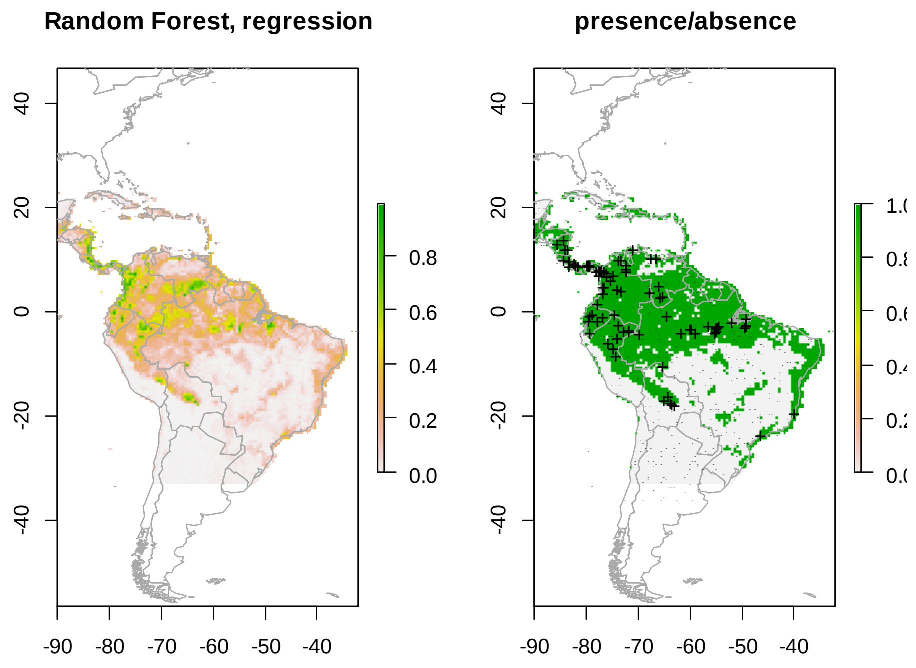
```
model <- pa ~ bio1 + bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17
rf1 <- randomForest(model, data=envtrain)
erf <- evaluate(testpres, testbackg, rf1)
erf
```

```
class          : ModelEvaluation
n presences    : 23
n absences     : 100
AUC            : 0.8052174
cor            : 0.4554547
max TPR+TNR at : 0.0708
```

Make a pair of plots, as before.

```
pr <- predict(predictors, rf1, ext=extb)
par(mfrow=c(1,2))
```

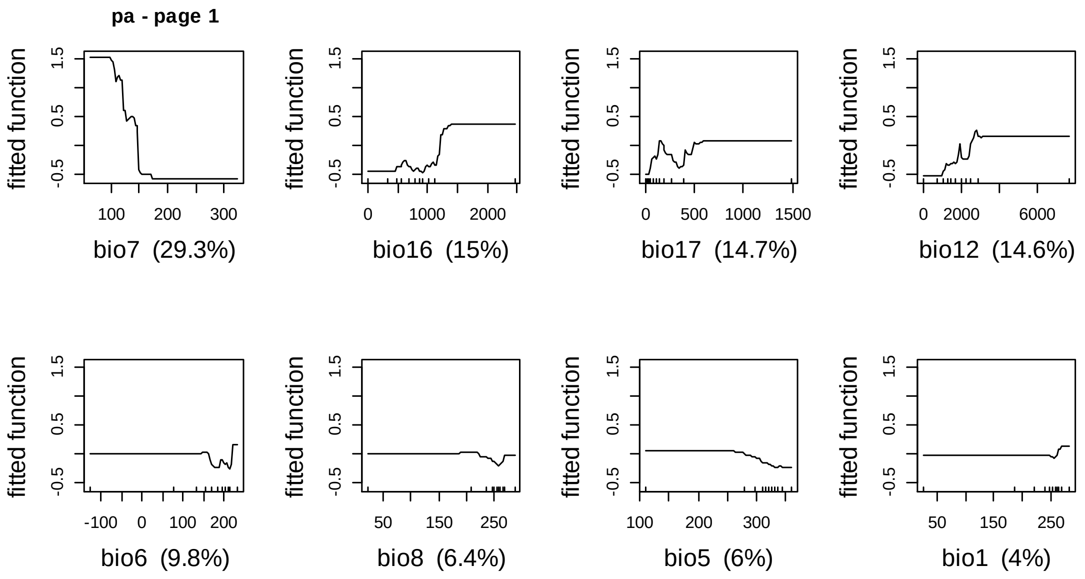
```
plot(pr, main='Random Forest, regression')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(erf, 'spec_sens')
plot(pr > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```

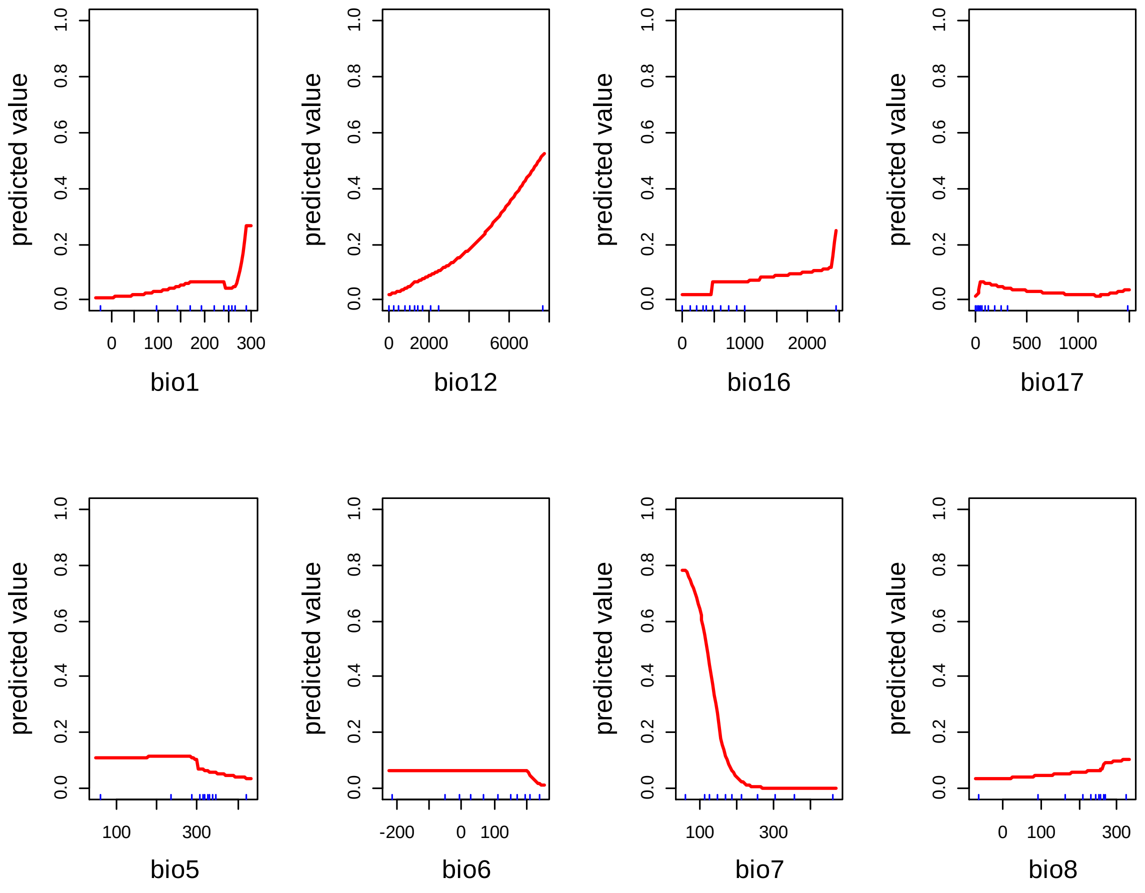


Some other packages can be used to fit boosted regression trees. The gbm package produces some helpful plots.
Below, in the call to the gbm.step function, gbm.y = 1 means the response variable (presence or absence) is in column 1. gbm.x = 2:9 means there are 8 predictors in columns 2 through 9.

```
rf2 <- gbm.step(data=envtrain, gbm.x = 2:9, gbm.y = 1,
            family = "bernoulli", tree.complexity = 5,
            learning.rate = 0.01, bag.fraction = 0.5)
summary(rf2)
gbm.plot(rf2)
```

(Top) Result of gbm.plot; (Bottom) Response plot from MaxEnt for comparison.


pa - page 1
fitted function
bio7 (29.3%)
bio16 (15%)
bio17 (14.7%)
bio12 (14.6%)
bio6 (9.8%)
bio8 (6.4%)
bio5 (6%)
bio1 (4%)


predicted value
bio1
bio12
bio16
bio17
bio5
bio6
bio7
bio8