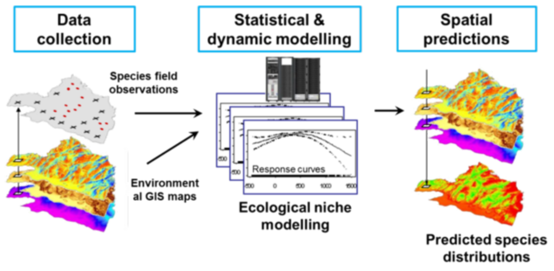
Tree-based Methods - GBM and niche modeling

Kevin Neal

February 11, 2016

What is Environmental Niche Modeling?

- statistically predict species distribution based on known occurrences and environmental data
- useful for:
  - predicting new localities
  - understanding biological niche space
  - predicting past and future distributions


Data collection
Statistical & dynamic modelling
Spatial predictions
Species field observations
Environmental GIS maps
Response curves
Ecological niche modelling
Predicted species distributions

- Spea hammondii - western spadefoot toad

Response variable: binary presence (1) or absence (0)

Predictor environmental variables:

- BIO1 = Annual Mean Temperature
- BIO2 = Mean Diurnal Range (Mean of monthly (max temp - min temp))
- BIO3 = Isothermality (BIO2/BIO7) (* 100)
- BIO4 = Temperature Seasonality (standard deviation *100)
- BIO5 = Max Temperature of Warmest Month
- BIO6 = Min Temperature of Coldest Month
- BIO7 = Temperature Annual Range (BIO5-BIO6)
- BIO8 = Mean Temperature of Wettest Quarter
- BIO9 = Mean Temperature of Driest Quarter
- BIO10 = Mean Temperature of Warmest Quarter
- BIO11 = Mean Temperature of Coldest Quarter
- BIO12 = Annual Precipitation
- BIO13 = Precipitation of Wettest Month
- BIO14 = Precipitation of Driest Month
- BIO15 = Precipitation Seasonality (Coefficient of Variation)
- BIO16 = Precipitation of Wettest Quarter
- BIO17 = Precipitation of Driest Quarter
- BIO18 = Precipitation of Warmest Quarter
- BIO19 = Precipitation of Coldest Quarter

Classification Trees and Environmental Niche Modeling of *Spea hammondii*

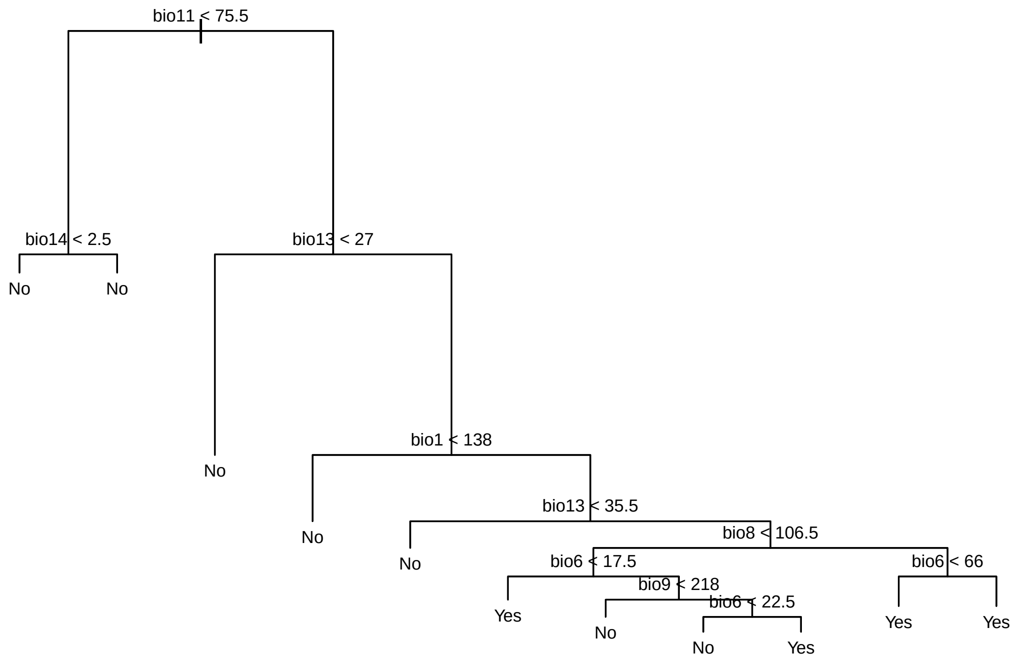
Single Classification Tree with Pruning

```
library(tree)
PresYN <- ifelse(present<=0.5, "No", "Yes") # converts Sales to binary Yes or No
speaYN <- data.frame(spea, PresYN)

tree.spea <- tree(PresYN~. -lat-lon-present, speaYN)
```

```
library(tree)
summary(tree.spea)
```

```
##
## Classification tree:
## tree(formula = PresYN ~ . - present - lon - lat, data = speaYN,
##     subset = train)
## Variables actually used in tree construction:
## [1] "bio11" "bio14" "bio13" "bio1"  "bio8"  "bio6"  "bio9"
## Number of terminal nodes:  11
## Residual mean deviance:  0.3824 = 46.27 / 121
## Misclassification error rate: 0.07576 = 10 / 132
```

```
plot(tree.spea)
text(tree.spea,pretty=0,cex=0.7)
```



```r
#tree.spea
```

```r
# split observations into training and test sets,
# build tree using training set, evaluate performance using test data
set.seed(2)
train.tree.spea <- sample(1:nrow(speaYN), 132)
test.tree.spea <- speaYN[-train,]
PresYN.test <- PresYN[-train.tree.spea]
tree.spea <- tree(PresYN~.-present-lon-lat,speaYN,subset=train)
tree.spea.pred <- predict(tree.spea, test.tree.spea, type="class")
```

```r
table(tree.spea.pred, PresYN.test)
```

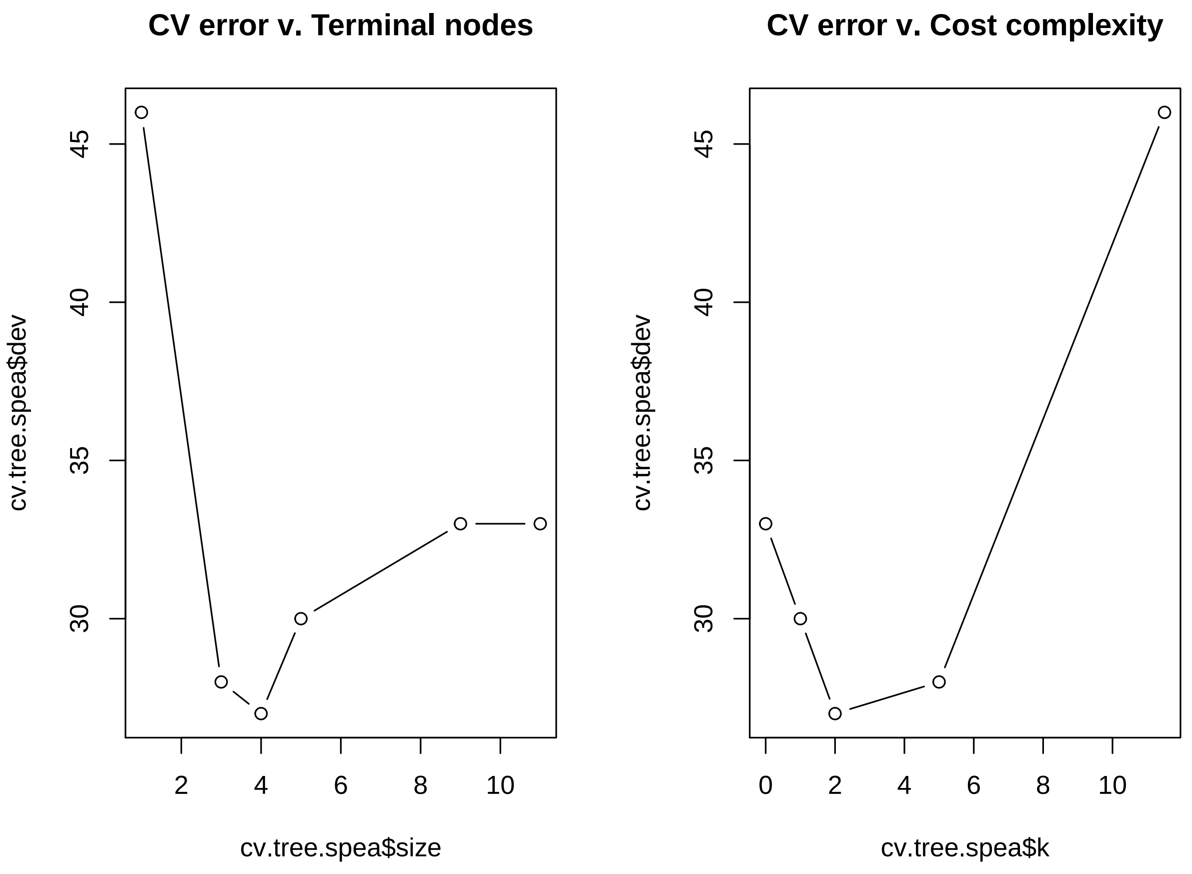
```
##                PresYN.test
## tree.spea.pred No Yes
##            No  75   3
##            Yes 14  40
```

```r
(80+36)/132 # rate of correct predictions
```

```
## [1] 0.8787879
```

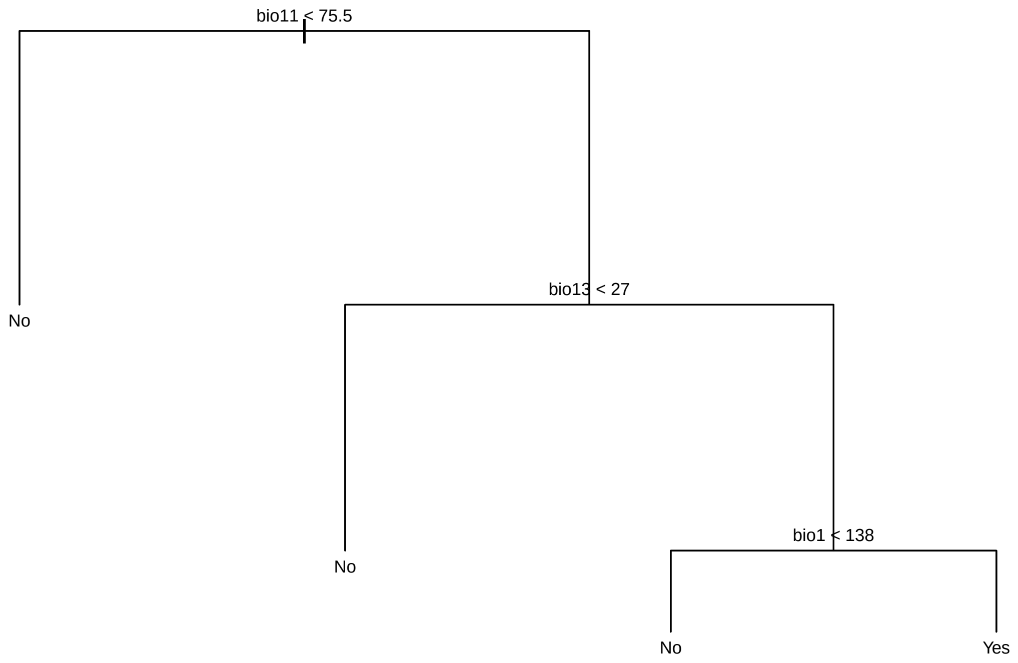
```r
# cv.tree() performs cross-validation to determine optimal level of tree complexity
set.seed(3)
cv.tree.spea <- cv.tree(tree.spea, FUN=prune.misclass)
names(cv.tree.spea)
cv.tree.spea # k refers to the cost-complexity parameter
# dev refers to cross-validation error rate in this instance
# tree with 4 nodes has lowest CV-error rate, at 27
```

```r
par(mfrow=c(1,2))
plot(cv.tree.spea$size, cv.tree.spea$dev,
     type="b", main="CV error v. Terminal nodes")
plot(cv.tree.spea$k, cv.tree.spea$dev,
     type="b", main="CV error v. Cost complexity")
```



```r
# prune tree to 4 terminal nodes

prune.tree.spea <- prune.misclass(tree.spea, best=4)
```

```r
par(mfrow=c(1,1))
plot(prune.tree.spea)
text(prune.tree.spea, pretty=0, cex=0.7) # returns bio11, bio13, and bio1
```



```r
summary(prune.tree.spea)
```

```
##
## Classification tree:
## snip.tree(tree = tree.spea, nodes = c(2L, 15L))
## Variables actually used in tree construction:
## [1] "bio11" "bio13" "bio1"
## Number of terminal nodes:  4
## Residual mean deviance:  0.594 = 76.03 / 128
## Misclassification error rate: 0.1212 = 16 / 132
```

```r
tree.spea.pred <- predict(prune.tree.spea,
                          test.tree.spea,
                          type="class")

table(tree.spea.pred, PresYN.test)
```

```
##                 PresYN.test
## tree.spea.pred No Yes
##            No  75   3
##            Yes 14  40
```

```
(75+40)/132
```

```
## [1] 0.8712121
```

```
# 4-node dataset has as good or slightly lower prediction rate as the unpruned
```

Boosted Regression Tree method of Elith et al

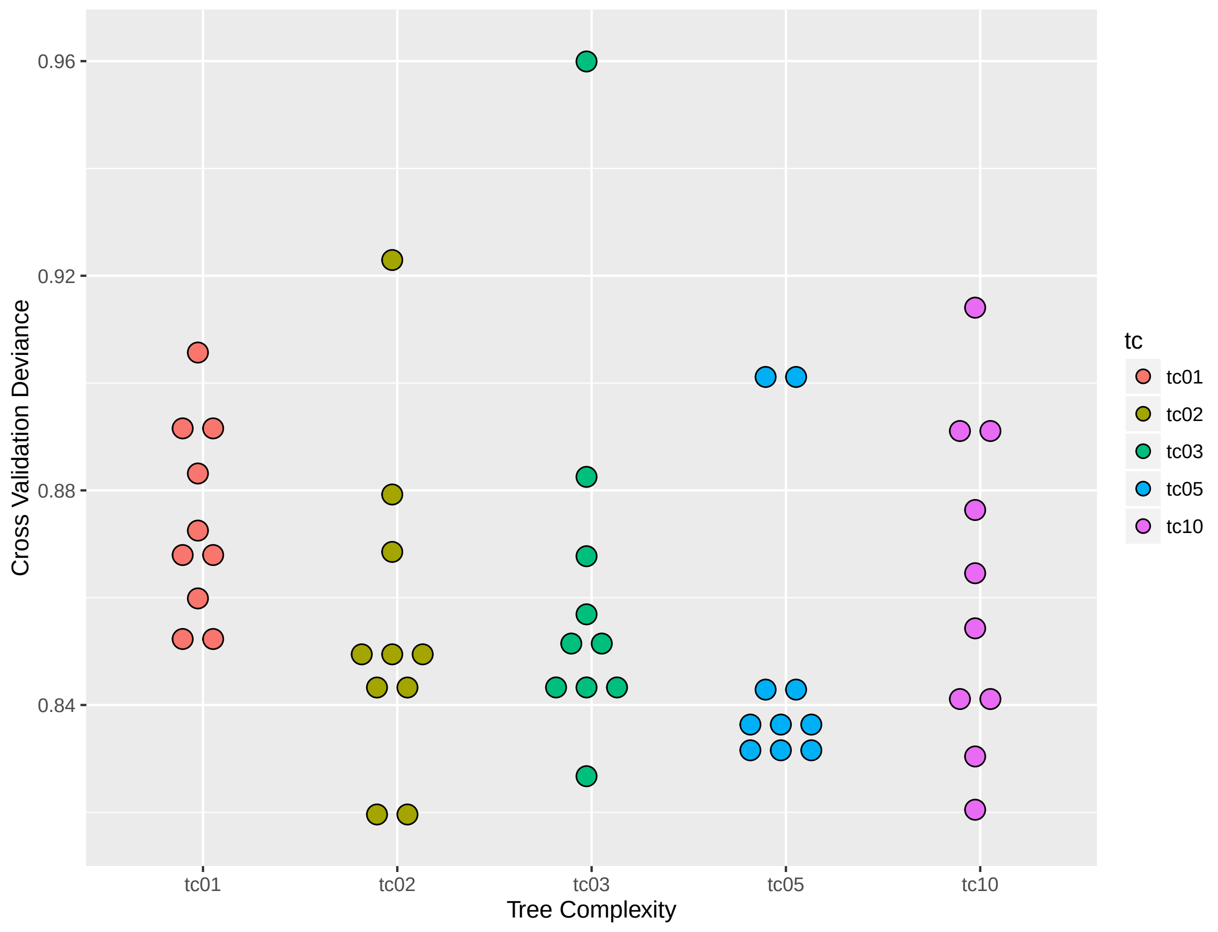
```r
library(dismo)
# https://cran.r-project.org/web/packages/dismo/vignettes/brt.pdf

spea_train <- spea[train,]
spea_test <- spea[-train,]


# identify optimal number of trees
# tweak parameters

# 5-fold CV seems to work better than 10 here...
cvdev <- cbind(rep(NA, 10),rep(NA, 10), rep(NA, 10), rep(NA, 10), rep(NA, 10))
colnames(cvdev) <- c("tc01", "tc02", "tc03", "tc05", "tc10")
tc <- c(1,2,3,5,10)
for (j in 1:5){
  for (i in 1:10){
    mm <- gbm.step(data=spea_train,
                   gbm.x=4:22,
                   gbm.y=1,
                   family="bernoulli",
                   tree.complexity=tc[j],
                   learning.rate=0.001,
                   bag.fraction=0.5,
                   n.folds=5,
                   silent=T)
    #cvdev 0.73, cvAUC 0.908
    cvdev[i,j] <- mm$cv.statistics$deviance.mean
  }
}
#plot as dotplot... use ggplot and tidyr cuz i screwed up the format to do it right

cvdev.long <- gather(as.data.frame(cvdev), "tc", "cvdev", 1:5)

cvdevplot <- ggplot(cvdev.long, aes(x=tc, y=cvdev, fill=tc)) +
  geom_dotplot(binaxis="y", stackdir="center",
               stackratio=1.5, dotsize=0.8)
cvdevplot + theme_classic() # tree complexity of 5 looks best here

bestmodel <- gbm.step(data=spea_train,
                      gbm.x=4:22,
                      gbm.y=1,
                      family="bernoulli",
                      tree.complexity=5,
                      learning.rate=0.001,
                      bag.fraction=0.5,
                      n.folds=5,
                      silent=T)

spea.simp <- gbm.simplify(bestmodel, n.drops=10)

bestmodel.simp <- gbm.step(data=spea_train,
                           gbm.x=spea.simp$pred.list[[8]],
                           gbm.y=1,
                           family="bernoulli",
                           tree.complexity=5,
                           learning.rate=0.001,
                           bag.fraction=0.5,
                           n.folds=5)
#dropping 8 is best for reducing error

bestmodel$cv.statistics$deviance.mean
bestmodel.simp$cv.statistics$deviance.mean
```

```r
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

```r
cvdevplot + labs(x="Tree Complexity", y="Cross Validation Deviance")
```

```
## `stat_bindot()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
# tree complexity of 5 looks best here

bestmodel$cv.statistics$deviance.mean
```

```
## [1] 0.8608191
```

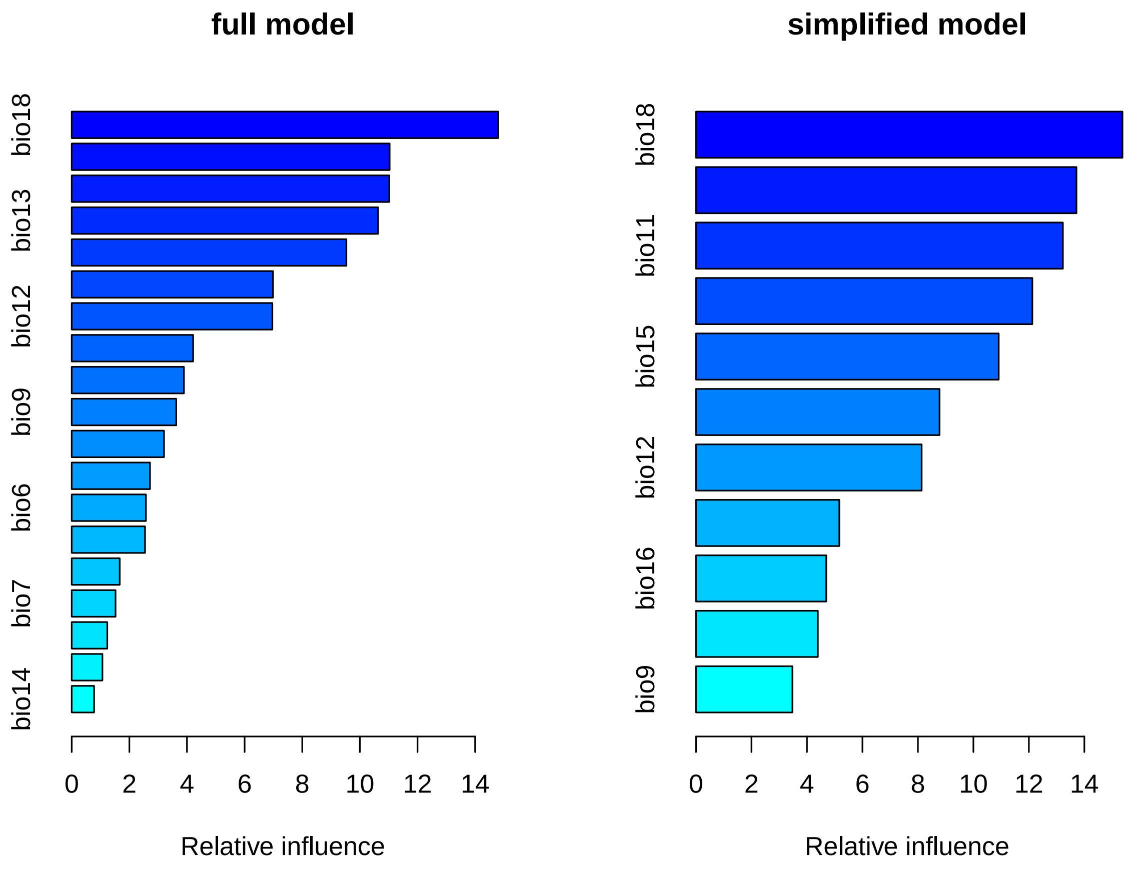
```r
bestmodel.simp$cv.statistics$deviance.mean
```

```
## [1] 0.8390673
```

```r
par(mfrow=c(1,2))
summary(bestmodel, main="full model")
```

```
##         var   rel.inf
## bio18 bio18 14.799994
## bio11 bio11 11.032991
## bio1   bio1 11.021977
## bio13 bio13 10.630306
## bio15 bio15  9.532248
## bio8   bio8  6.985348
## bio12 bio12  6.963616
## bio10 bio10  4.213796
## bio16 bio16  3.893484
## bio9   bio9  3.627175
## bio19 bio19  3.202177
## bio4   bio4  2.715925
## bio6   bio6  2.574758
## bio2   bio2  2.543840
## bio5   bio5  1.665150
## bio7   bio7  1.520659
## bio17 bio17  1.234143
## bio3   bio3  1.065813
## bio14 bio14  0.776600
```

```
summary(bestmodel.simp, main="simplified model")
```



```
##         var   rel.inf
## bio18 bio18 15.374319
## bio13 bio13 13.715635
## bio11 bio11 13.226314
## bio1   bio1 12.125256
## bio15 bio15 10.912943
## bio8   bio8  8.779849
## bio12 bio12  8.135073
## bio10 bio10  5.167207
## bio16 bio16  4.695085
## bio4   bio4  4.392838
## bio9   bio9  3.475481
```

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

Both models show highest importance for BIO18 (precipitation of warmest quarter)

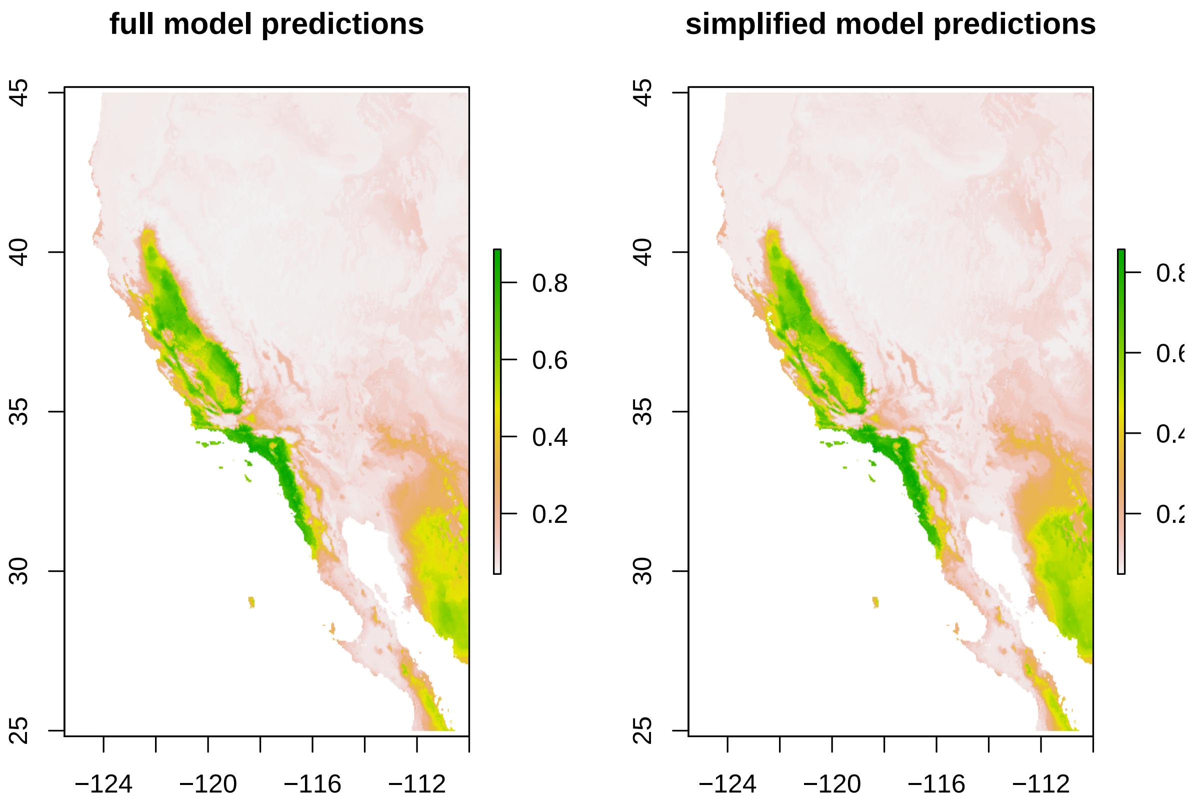
Full model top 5:
- bio18 (precipitation of warmest quarter)
- bio11 (mean temp of coldest quarter)
- bio1 (annual mean temperature)
- bio13 (precipitation of wettest month)
- bio15 (precipitation seasonality [coefficient of variation])

Simplified model top 5: bio18, bio13, bio11, bio1, bio15

Pruned tree: bio11, bio13, bio1

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

library(dismo)
library(raster)

bclimRaster <- brick("C:/Users/Kevin/Google Drive/UCLA Courses or Lab meetings etc/EEB 234/Final Projec

bclimRaster.simp <- bclimRaster[[c(spea.simp$pred.list[[8]]-3)]]
# exclude layers eliminated in the simplified model by gbm.simplify()

spea.predict <- predict(bclimRaster,
                        model,
                        n.trees=model$gbm.call$best.trees,
                        type="response")

spea.predict.simp <- predict(bclimRaster.simp,
                             bestmodel.simp,
                             n.trees=bestmodel.simp$gbm.call$best.trees,
                             type="response")

# type="response" gives probabilities on logit scale
# using the response variable, i.e. presence/absence
```

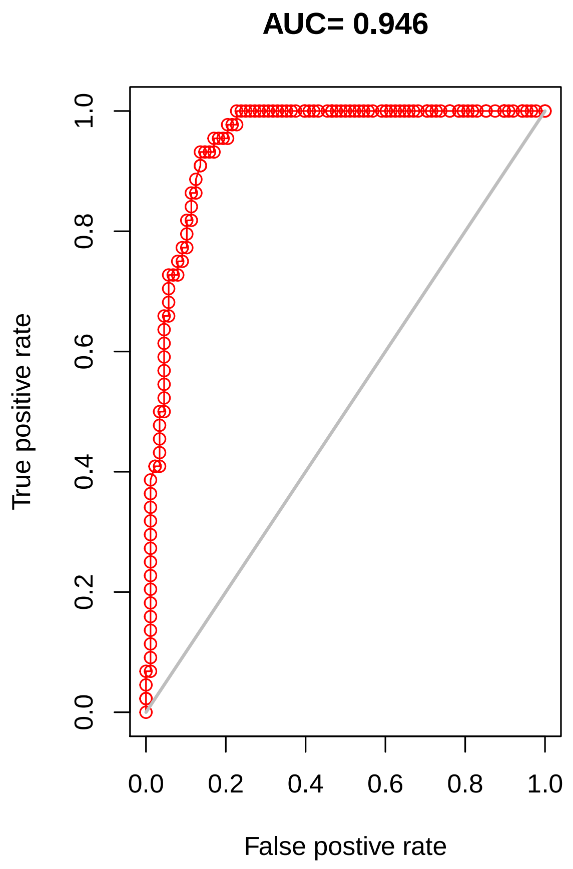
```r
par(mfrow=c(1,2))
plot(spea.predict,
     main="full model predictions")
plot(spea.predict.simp,
     main="simplified model predictions")
```

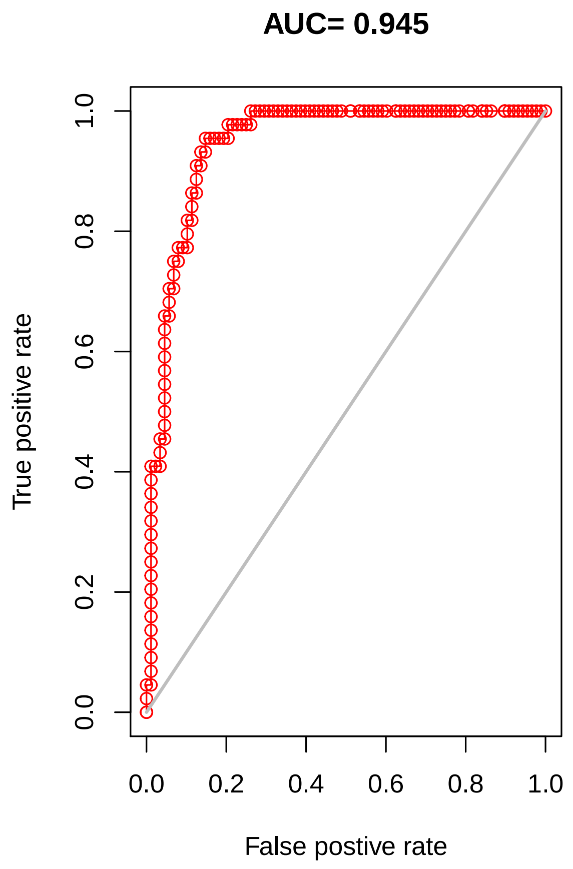

full model predictions
simplified model predictions
45
40
35
30
25
−124
−120
−116
−112
0.8
0.6
0.4
0.2

Model evaluation

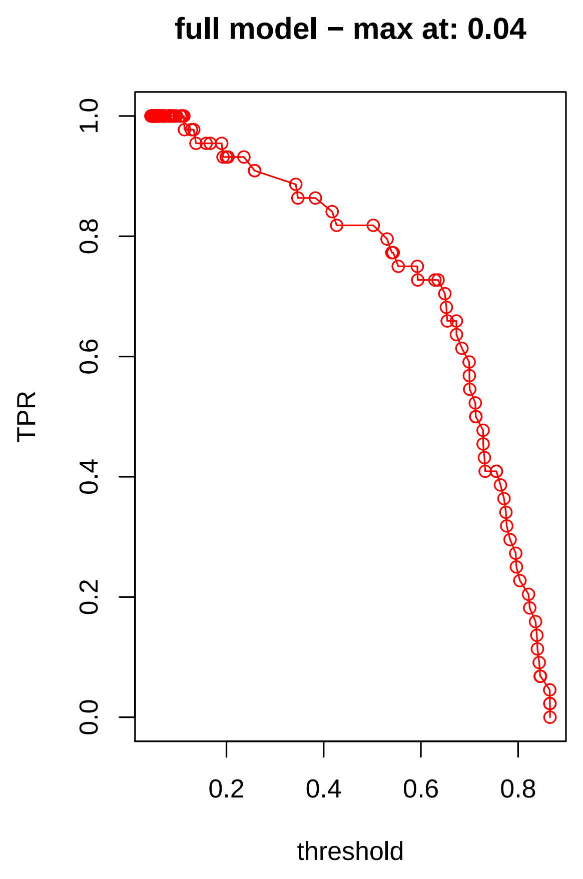
```r
ee <- evaluate(
  p=spea_test[spea_test[,1]==1,2:3],
  a=spea_test[spea_test[,1]==0,2:3],
  model=model,
  n.trees=model$gbm.call$best.trees,
  type="response",
  x=bclimRaster)
# evaluates model using sample coordinates to get predictor values
# from raster brick and then calculate the response
# (i.e. predicted presence probability)

ee.simp <- evaluate(
  p=spea_test[spea_test[,1]==1,2:3],
  a=spea_test[spea_test[,1]==0,2:3],
  model=model,
  n.trees=bestmodel.simp$gbm.call$best.trees,
  type="response",
  x=bclimRaster)

# ee <- evaluate(p=spea[1:88,2:3],
# a=spea[89:264,2:3],
# model=model,
# n.trees=model$gbm.call$best.trees,
# type="response",
# x=bclimRaster)
# run this if you want to evaluate using all data points for some reason
```

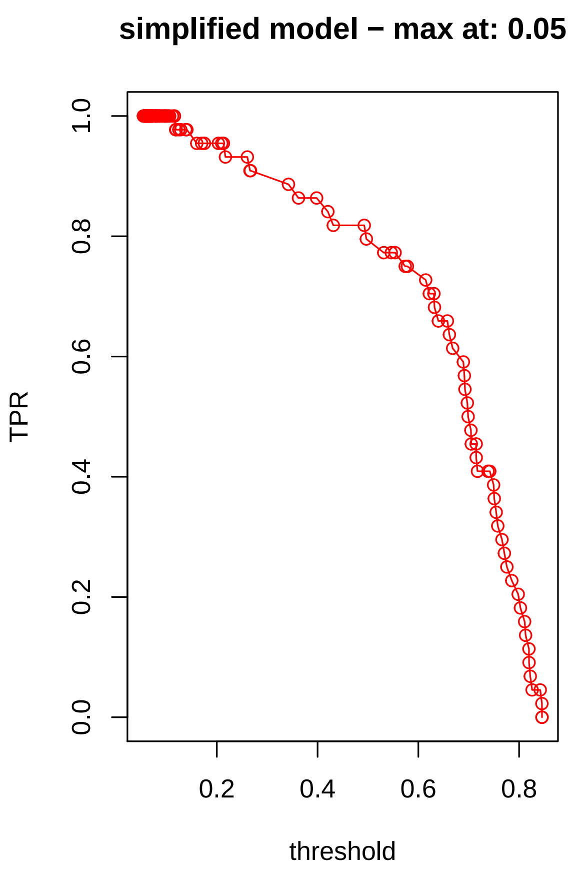
```r
par(mfrow=c(1,2))

plot(ee, "ROC", main="full model")
plot(ee.simp, "ROC", main="simplified model")
```



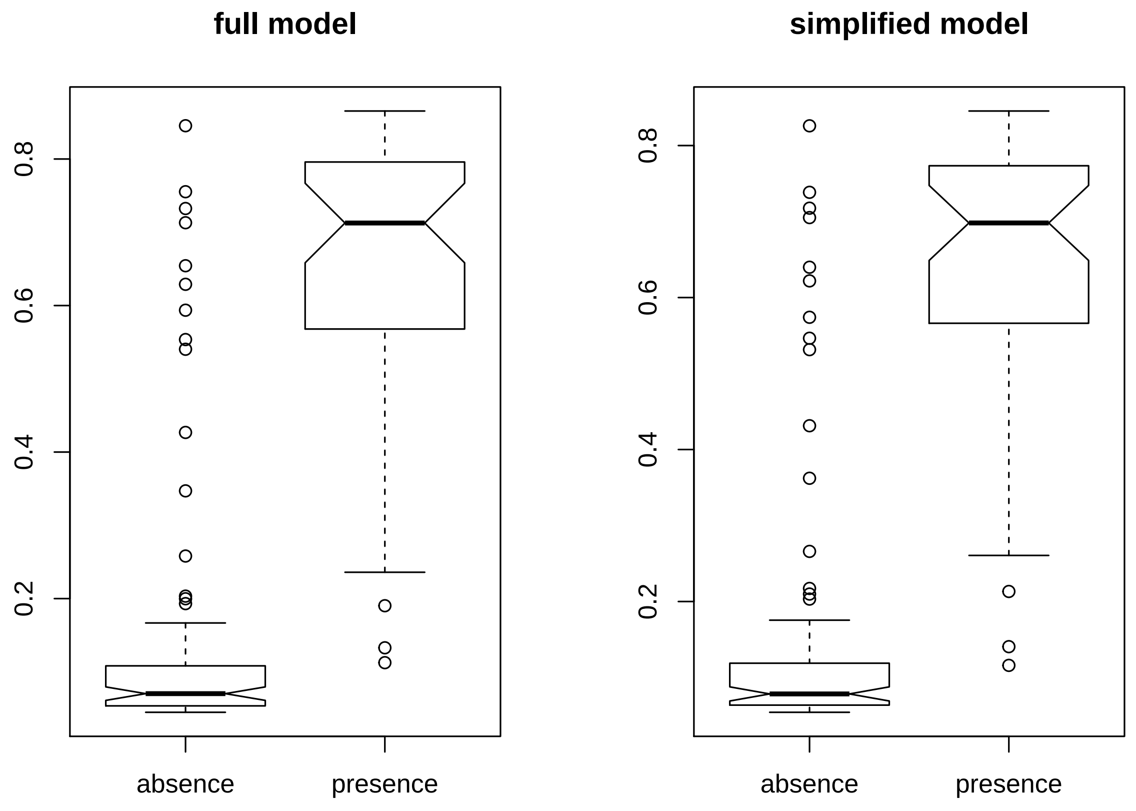


```r
plot(ee, "TPR", main="full model")
plot(ee.simp, "TPR", main="simplified model")
```

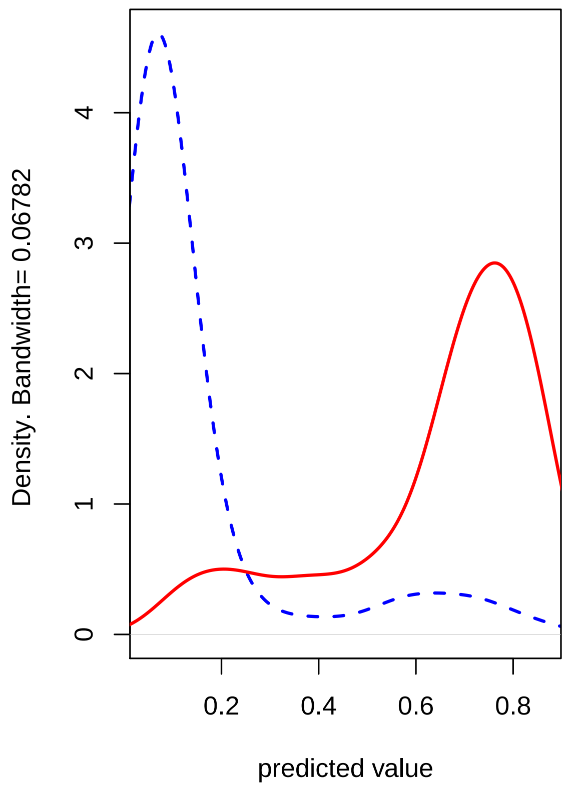


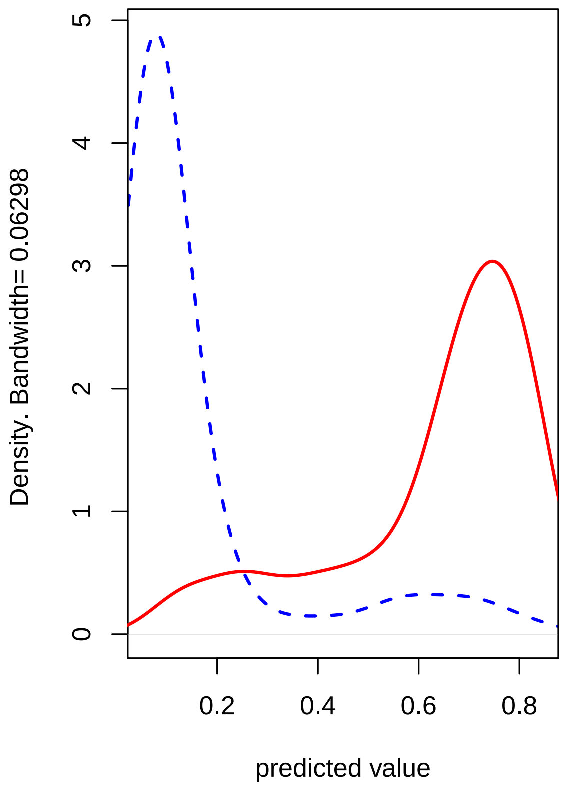


```
boxplot(ee, main="full model")
boxplot(ee.simp, main="simplified model")
```



```
density(ee)
density(ee.simp)
```


Density. Bandwidth= 0.06782
0
1
2
3
4
0.2
0.4
0.6
0.8
predicted value


Density. Bandwidth= 0.06298
0
1
2
3
4
5
0.2
0.4
0.6
0.8
predicted value