Boosted Regression Trees for ecological modeling

Jane Elith and John Leathwick

August 2, 2010

1 Introduction

This is a brief tutorial to accompany a set of functions that we have written to facilitate fitting BRT (boosted regression tree) models in R. This tutorial is a modified version of the tutorial accompaniying Elith, Leathwick and Hastie's article in Journal of Animal Ecology. It has been adjusted to match the implementation of these functions in the 'dismo' package. The gbm* functions in the dismo package extend functions in the 'gbm' package by Greg Ridgeway. The goal of our functions is to make the functions in the 'gbm' package easier to apply to ecological data, and to enhance interpretation.

The tutorial is aimed at helping you to learn the mechanics of how to use the functions and to develop a BRT model in R. It does not explain what a BRT model is - for that, see the references at the end of the tutorial, and the documentation of the gbm package. For an example application with similar data as in this tutorial, see Elith et al., 2008.

The gbm functions in 'dismo' are as follows:

1. gbm.step - Fits a gbm model to one or more response variables, using cross-validation to estimate the optimal number of trees. This requires use of the utility functions roc, calibration and calc.deviance.

2. gbm.fixed, gbm.holdout - Alternative functions for fitting gbm models, implementing options provided in the gbm package.

3. gbm.simplify - Code to perform backwards elimination of variables, to drop those that give no evidence of improving predictive performance.

4. gbm.plot - Plots the partial dependence of the response on one or more predictors.

5. gbm.plot.fits - Plots the fitted values from a gbm object returned by any of the model fitting options. This can give a more reliable guide to the shape of the fitted surface than can be obtained from the individual functions, particularly when predictor variables are correlated and/or samples are unevenly distributed in environmental space.

6. gbm.interactions - Tests whether interactions have been detected and modelled, and reports the relative strength of these. Results can be visualised with gbm.perspec

2 Example data

Two sets of presence/absence data for Anguilla australis (Angaus) are available. One for model training (building) and one for model testing (evaluation). In the example below we load the training data. Presence (1) and absence (0) is recorded in column 2. The environmental variables are in columns 3 to 14. This is the same data as used in Elith, Leathwick and Hastie (2008).

```
> library(dismo)

raster version 1.3-7 (2-August-2010)

> data(Anguilla_train)
> head(Anguilla_train)

  Site Angaus SegSumT SegTSeas SegLowFlow DSDist DSMaxSlope USAvgT USRainDays
1    1      0    16.0    -0.10      1.036  50.20       0.57   0.09      2.470
2    2      1    18.7     1.51      1.003 132.53       1.15   0.20      1.153
3    3      0    18.3     0.37      1.001 107.44       0.57   0.49      0.847
4    4      0    16.7    -3.80      1.000 166.82       1.72   0.90      0.210
5    5      1    17.2     0.33      1.005   3.95       1.15  -1.20      1.980
6    6      0    15.1     1.83      1.015  11.17       1.72  -0.20      3.300
  USSlope USNative DSDam   Method LocSed
1     9.8     0.81     0 electric    4.8
2     8.3     0.34     0 electric    2.0
3     0.4     0.00     0      spo    1.0
4     0.4     0.22     1 electric    4.0
5    21.9     0.96     0 electric    4.7
6    25.7     1.00     0 electric    4.5

> Anguilla_train = Anguilla_train[1:250, ]
```

3 Fitting a model

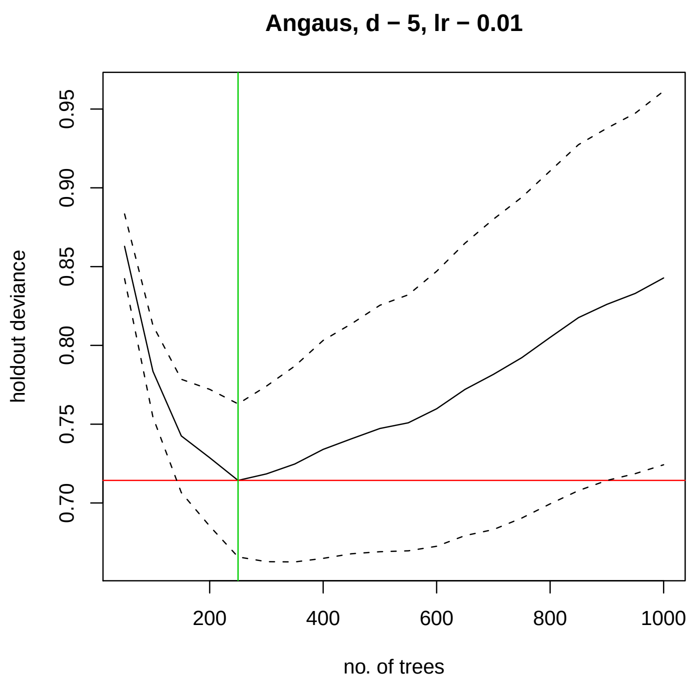
To fit a gbm model, you need to decide what settings to use the article associated with this tutorial gives you information on what to use as rules of thumb. These data have 1000 sites, comprising 202 presence records for the short-finned eel (the command sum(model.data$Angaus)will give you the total number of presences). As a first guess you could decide: 1. There are enough data to model interactions of reasonable complexity 2. A lr of about 0.01 could be a reasonable starting point.

To below example shows how to use our function that steps forward and identifies the optimal number of trees (nt).

```
> angaus.tc5.lr01 <- gbm.step(data = Anguilla_train, gbm.x = 3:13,
+     gbm.y = 2, family = "bernoulli", tree.complexity = 5, learning.rate = 0.01,
+     bag.fraction = 0.5)
```

```
Loaded gbm 1.6-3.1


 GBM STEP - version 2.9

Performing cross-validation optimisation of a boosted regression tree model
for Angaus with dataframe Anguilla_train and using a family of bernoulli
Using 250 observations and 11 predictors
creating 10 initial models of 50 trees

 folds are stratified by prevalence
total mean deviance =  1.0436
tolerance is fixed at  0.001
ntrees resid. dev.
50    0.8667
now adding trees...
100   0.7814
150   0.7358
200   0.7136
250   0.696
300   0.688
350   0.6885
400   0.6864
450   0.6888
500   0.6941
550   0.6982
600   0.7032
650   0.7056
700   0.7092
750   0.7128
800   0.7168
850   0.7243
900   0.7308
950   0.7381
1000   0.7451
1050   0.7525
fitting final gbm model with a fixed number of  400  trees for  Angaus

mean total deviance = 1.044
mean residual deviance = 0.328

estimated cv deviance = 0.686 ; se = 0.076

training data correlation = 0.888
cv correlation =  0.613 ; se = 0.06
```

```
training data ROC score = 0.992
cv ROC score = 0.878 ; se = 0.024

elapsed time -  0.1 minutes
```



Above we used the function gbm.step, this function is an alternative to the cross-validation provided in the gbm package. We have passed information to the function about data and settings. We have defined:

the dataframe containing the data = Anguilla_train;

the predictor variables - gbm.x = c(3:13) - which we do using a vector consisting of the indices for the data columns containing the predictors (i.e., here the predictors are columns 3 to 13 in Anguilla_train);

the response variable - gbm.y = 2 - indicating the column number for the species (response) data;

the nature of the error structure - For example, family = 'bernoulli' (note the quotes);

the tree complexity - we are trying a tree complexity of 5 for a start;

the learning rate - we are trying with 0.01;

the bag fraction - our default is 0.75; here we are using 0.5;

Everything else - i.e. all the other things that we could change if we wanted to (see the help file, and the documentation of the gbm package) - are set at their defaults if they are not named in the call. If you want to see what else you could change, you can type gbm.step and all the code will write itself to screen, or type args(gbm.step) and it will open in an editor window.

Running a model such as that described above writes progress reports to the screen, makes a graph, and returns an object containing a number of components. Firstly, the things you can see: The R console will show something like this (not identical, because remember that these models are stochastic and therefore slightly different each time you run them, unless you set the seed or make them deterministic by using a bag fraction of 1)

This reports a brief model summary. All these values are also retained in the model object, so they will be permanently kept (as long as you save the R workspace before quitting).

This model was built with the default 10-fold cross-validation. The solid black curve is the mean, and the dotted curves about 1 standard error, for the changes in predictive deviance (ie as measured on the excluded folds of the cross-validation). The red line shows the minimum of the mean, and the green line the number of trees at which that occurs. The final model that is returned in the model object is built on the full data set, using the number of trees identified as optimal.

The returned object is a list (see R documentation if you don't know what that is), and the names of the components can be seen by typing:

To pull out one component of the list, use a number (angaus.tc5.lr01[[29]]) or name (angaus.tc5.lr01$cv.statistics) - but be careful, some are as big as the dataset, e.g. there will be 1000 fitted values. Find this by typing length(angaus.tc5.lr01$fitted)

The way we organise our functions is to return exactly what Ridgeway's function in the gbm package returned, plus extra things that are relevant to our code. You will see by looking at the final parts of the gbm.step code that we have added components 25 onwards, that is, from gbm.call on. See the gbm documentation for what his parts comprise. Ours are:

gbm.call - A list containing the details of the original call to gbm.step

fitted - The fitted values from the final tree, on the response scale

fitted.vars - The variance of the fitted values, on the response scale

residuals - The residuals for the fitted values, on the response scale

contributions - The relative importance of the variables, produced from the gbm summary function

self.statistics - The relevant set of evaluation statistics, calculated on the fitted values - i.e. this is only interesting in so far as it demonstrates "evaluation" (i.e. fit) on the training data. It should NOT be reported as the model predictive performance.

cv.statistics These are the most appropriate evaluation statistics. We calculate each statistic within each fold (at the identified optimal number of trees that is calculated on the mean change in predictive deviance over all folds), then present here the mean and standard error of those fold-based statistics.

weights - the weights used in fitting the model (by default, "1" for each observation - i.e. equal weights).

trees.fitted - A record of the number of trees fitted at each step in the stagewise fitting; only relevant for later calculations training.loss.values - The stagewise changes in deviance on the training data

cv.values - the mean of the CV estimates of predictive deviance, calculated at each step in the stagewise process - this and the next are used in the plot shown above

cv.loss.ses - standard errors in CV estimates of predictive deviance at each step in the stagewise process

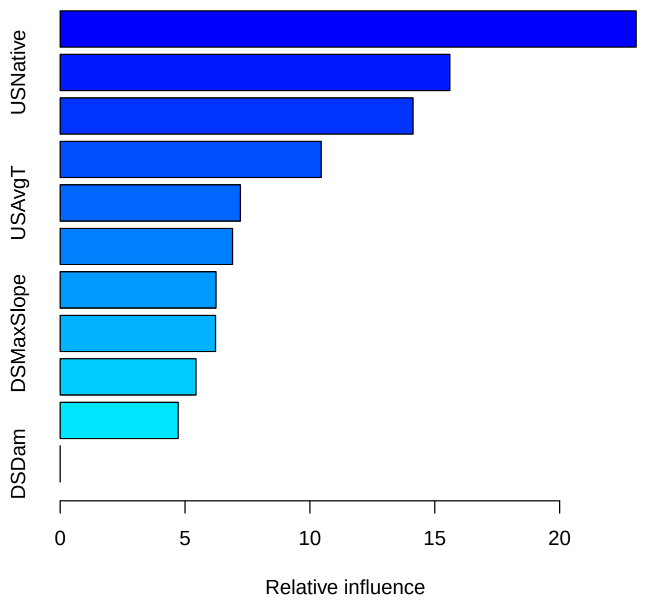
cv.loss.matrix - the matrix of values from which cv.values were calculated - as many rows as folds in the CV

cv.roc.matrix - as above, but the values in it are area under the curve estimated on the excluded data, instead of deviance in the cv.loss.matrix.

You can look at variable importance with the summary function

```
> names(angaus.tc5.lr01)

 [1] "initF"                "fit"                  "train.error"
 [4] "valid.error"          "oobag.improve"        "trees"
 [7] "c.splits"             "bag.fraction"         "distribution"
[10] "interaction.depth"    "n.minobsinnode"       "n.trees"
[13] "nTrain"               "response.name"        "shrinkage"
[16] "train.fraction"       "var.levels"           "var.monotone"
[19] "var.names"            "var.type"             "verbose"
[22] "data"                 "Terms"                "cv.folds"
[25] "gbm.call"             "fitted"               "fitted.vars"
[28] "residuals"            "contributions"        "self.statistics"
[31] "cv.statistics"        "weights"              "trees.fitted"
[34] "training.loss.values" "cv.values"            "cv.loss.ses"
[37] "cv.loss.matrix"       "cv.roc.matrix"
```

```
> summary(angaus.tc5.lr01)

          var   rel.inf
1     SegSumT 23.065058
2    USNative 15.604541
3      Method 14.130890
4      DSDist 10.453492
5      USAvgT  7.217030
6  SegLowFlow  6.902896
7    SegTSeas  6.240838
8  DSMaxSlope  6.220863
9  USRainDays  5.438661
10    USSlope  4.725731
11      DSDam  0.000000
```



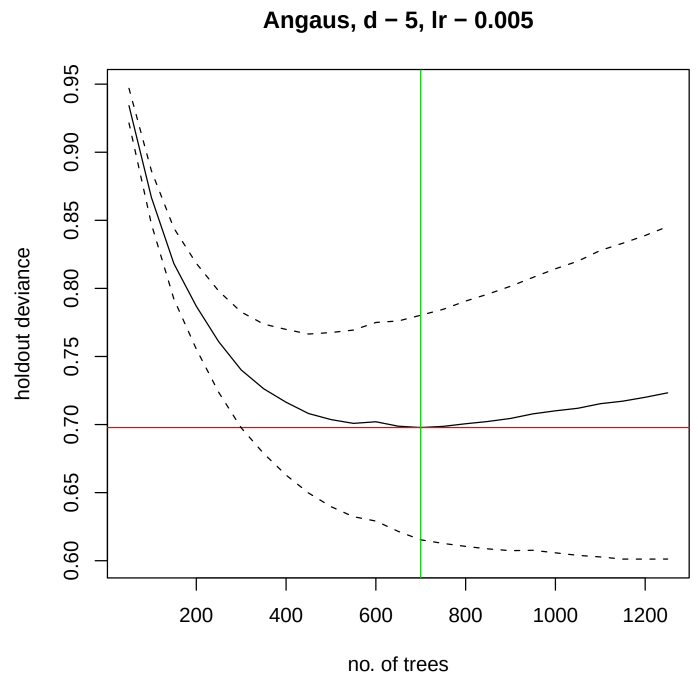
4 Choosing the settings

The above was a first guess at settings, using rules of thumb discussed in Elith et al. (2008). It made a model with only 650 trees, so our next step would be to reduce the lr. For example, try lr = 0.005, to aim for over 1000 trees:

```
> angaus.tc5.lr005 <- gbm.step(data = Anguilla_train, gbm.x = 3:13,
+     gbm.y = 2, family = "bernoulli", tree.complexity = 5, learning.rate = 0.005,
+     bag.fraction = 0.5)
```

```
 GBM STEP - version 2.9

Performing cross-validation optimisation of a boosted regression tree model
for Angaus with dataframe Anguilla_train and using a family of bernoulli
Using 250 observations and 11 predictors
creating 10 initial models of 50 trees

 folds are stratified by prevalence
total mean deviance =  1.0436
tolerance is fixed at  0.001
ntrees resid. dev.
50    0.9324
```

```
now adding trees...
100   0.8633
150   0.817
200   0.7838
250   0.7592
300   0.738
350   0.723
400   0.7139
450   0.7078
500   0.701
550   0.6985
600   0.699
650   0.6972
700   0.6971
750   0.7005
800   0.6992
850   0.7021
900   0.7055
950   0.7079
1000   0.711
1050   0.7168
1100   0.7189
1150   0.72
1200   0.7228
1250   0.7267
fitting final gbm model with a fixed number of  700  trees for  Angaus

mean total deviance = 1.044
mean residual deviance = 0.359

estimated cv deviance = 0.697 ; se = 0.077

training data correlation = 0.874
cv correlation =  0.613 ; se = 0.072

training data ROC score = 0.99
cv ROC score = 0.874 ; se = 0.033

elapsed time -  0.11 minutes
```



To more broadly explore whether other settings perform better, and assuming that these are the only data available, you could either split the data into a training and testing set or use the cross-validation results. You could systematically alter tc, lr and the bag fraction and compare the results. See the later section on prediction to find out how to predict to independent data and calculate relevant statistics.

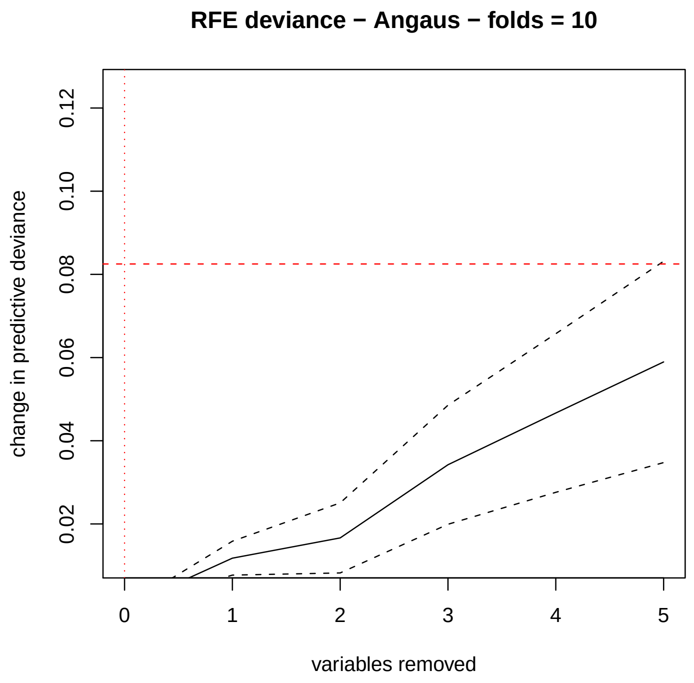
5 Alternative ways to fit models

The step function above is slower than just fitting one model and finding a minimum. If this is a problem, you could use our gbm.holdout code - this combines from the gbm package in ways we find useful. We tend to prefer gbm.step, especially when modelling many species, because it automatically finds the optimal number of trees. Alternatively, the gbm.fixed code allows you to fit a model of a set number of trees; this can be used, as in Elith et al. (2008), to predict to new data (see later section).

6 Simplifying the model

For a discussion of simplification see Appendix 2 of the online supplement to Elith et al (2008). Simplification builds many models, so it can be slow. For example, the code below took a few minutes to run on a modern laptop. In it we assess the value in simplifying the model built with a lr of 0.005, but only test dropping up to 5 variables (the ”n.drop” argument; the default is an automatic rule so it continues until the average change in predictive deviance exceeds its original standard error as calculated in gbm.step).

```
> angaus.simp <- gbm.simplify(angaus.tc5.lr005, n.drops = 5)

gbm.simplify - version 2.9

simplifying gbm.step model for Angaus with 11 predictors and 250 observations
original deviance = 0.6978(0.0825)

a fixed number of 5 drops will be tested

creating initial models...

dropping predictor:  1  2  3  4  5

now processing final dropping of variables with full data

1 - DSDam
2 - USRainDays
3 - USAvgT
4 - USSlope
5 - DSMaxSlope
```

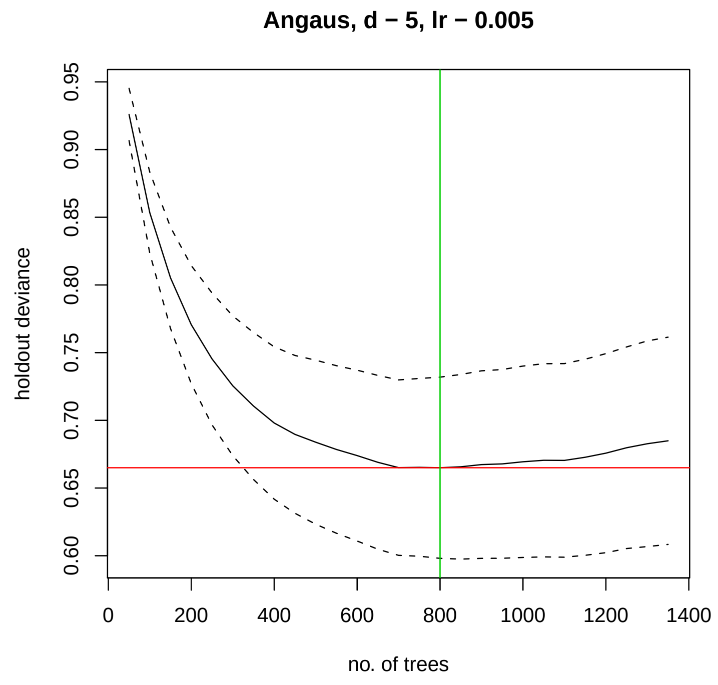


For our run, this estimated that the optimal number of variables to drop was 1; yours could be slightly different:

You can use the number indicated by the red vertical line, or look at the results in the angaus.simp object Now make a model with 1 predictor dropped, by indicating to the gbm.step call the relevant number of predictor(s) from the predictor list in the angaus.simp object - see highlights, below, in which we indicate we want to drop 1 variable by calling the second vector of predictor columns in the pred list, using a [[1]]:

```
> angaus.tc5.lr005.simp <- gbm.step(Anguilla_train, gbm.x = angaus.simp$pred.list[[1]],
+     gbm.y = 2, tree.complexity = 5, learning.rate = 0.005)

 GBM STEP - version 2.9

Performing cross-validation optimisation of a boosted regression tree model
for Angaus with dataframe Anguilla_train and using a family of bernoulli
Using 250 observations and 10 predictors
creating 10 initial models of 50 trees

 folds are stratified by prevalence
total mean deviance =  1.0436
tolerance is fixed at  0.001
ntrees resid. dev.
```

```
50    0.9217
now adding trees...
100   0.8476
150   0.7982
200   0.7651
250   0.7379
300   0.721
350   0.7075
400   0.699
450   0.6917
500   0.6886
550   0.6867
600   0.6864
650   0.6856
700   0.6869
750   0.6893
800   0.6904
850   0.6928
900   0.695
950   0.6985
1000   0.7024
1050   0.7057
1100   0.7082
1150   0.7123
1200   0.7158
fitting final gbm model with a fixed number of  650  trees for  Angaus

mean total deviance = 1.044
mean residual deviance = 0.337

estimated cv deviance = 0.686 ; se = 0.06

training data correlation = 0.899
cv correlation =  0.615 ; se = 0.043

training data ROC score = 0.994
cv ROC score = 0.888 ; se = 0.019

elapsed time -  0.1 minutes
```

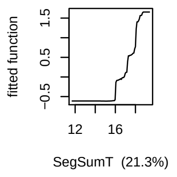


This has now made a new model (angaus.tc5.lr005.simp) with the same list components as described earlier. We could continue to use it, but given that we don't particularly want a more simple model (our view is that, in a dataset of this size, included variables that contribute little are acceptable), we won't use it further.

7 Plotting the functions and fitted values from the model

The fitted functions from a BRT model created from any of our functions can be plotted using gbm.plot. If you want to plot all variables on one sheet first set up a graphics device with the right set-up - here we will make one with 3 rows and 4 columns:

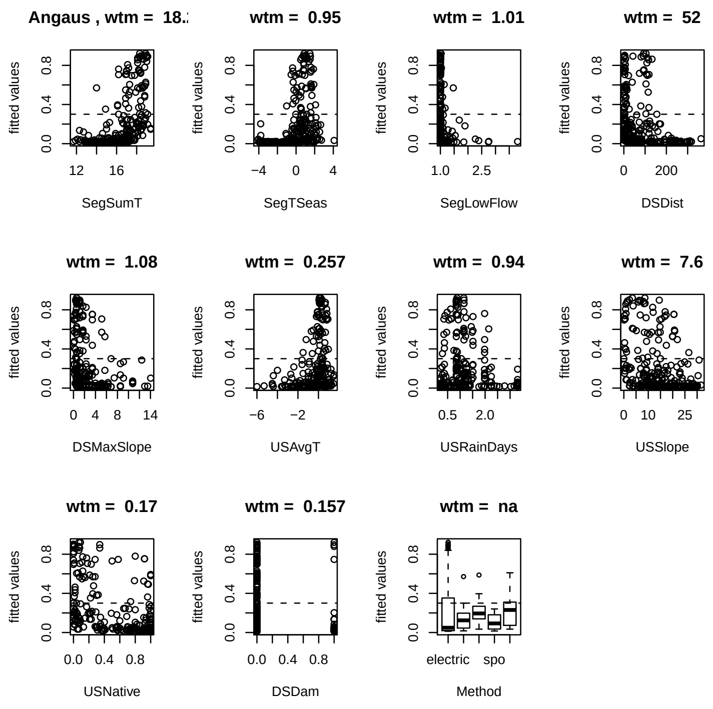
```
> gbm.plot(angaus.tc5.lr005, n.plots = 11, write.title = FALSE)
```



Additional arguments to this function allow for making a smoothed representation of the plot, allowing different vertical scales for each variable, omitting (and formatting) the rugs, and plotting a single variable.

Depending on the distribution of observations within the environmental space, fitted functions can give a misleading indication about the distribution of the fitted values in relation to each predictor. The function gbm.plot.fits has been provided to plot the fitted values in relation to each of the predictors used in the model.

```
> gbm.plot.fits(angaus.tc5.lr005)
```



This has options that allow for the plotting of all fitted values or of fitted values only for positive observations, or the plotting of fitted values in factor type graphs that are much quicker to print. Values above each graph indicate the weighted mean of fitted values in relation to each non-factor predictor.

8 Interrogate and plot the interactions

This code assesses the extent to which pairwise interactions exist in the data. find.int <- gbm.interactions(angaus.tc5.lr005). The returned object, here named test.int, is a list. The first 2 components summarise the results, first as a ranked list of the 5 most important pairwise interactions, and the second tabulating all pairwise interactions. The variable index numbers in $rank.list can be used for plotting.

```
> find.int <- gbm.interactions(angaus.tc5.lr005)
```

```
gbm.interactions - version 2.9
Cross tabulating interactions for gbm model with 11 predictors
1  2  3  4  5  6  7  8  9  10
```

```
> find.int$interactions
```

```
          SegSumT SegTSeas SegLowFlow DSDist DSMaxSlope USAvgT USRainDays
SegSumT         0     2.78       3.01   1.34       0.32  10.71       0.17
```

```
SegTSeas         0     0.00        0.42   2.86        0.09   0.11        4.97
SegLowFlow       0     0.00        0.00   0.21        0.08   0.09        0.06
DSDist           0     0.00        0.00   0.00        0.03   1.85        0.23
DSMaxSlope       0     0.00        0.00   0.00        0.00   0.08        0.11
USAvgT           0     0.00        0.00   0.00        0.00   0.00        1.17
USRainDays       0     0.00        0.00   0.00        0.00   0.00        0.00
USSlope          0     0.00        0.00   0.00        0.00   0.00        0.00
USNative         0     0.00        0.00   0.00        0.00   0.00        0.00
DSDam            0     0.00        0.00   0.00        0.00   0.00        0.00
Method           0     0.00        0.00   0.00        0.00   0.00        0.00
           USSlope USNative DSDam Method
SegSumT       3.34    23.38     0   5.16
SegTSeas      0.03     4.11     0   1.65
SegLowFlow    0.43     1.60     0   2.48
DSDist        0.07     1.93     0   1.11
DSMaxSlope    0.23     9.42     0   0.08
USAvgT        0.09     0.82     0   1.12
USRainDays    0.06     2.60     0   0.50
USSlope       0.00     0.86     0   0.04
USNative      0.00     0.00     0   8.42
DSDam         0.00     0.00     0   0.00
Method        0.00     0.00     0   0.00
```

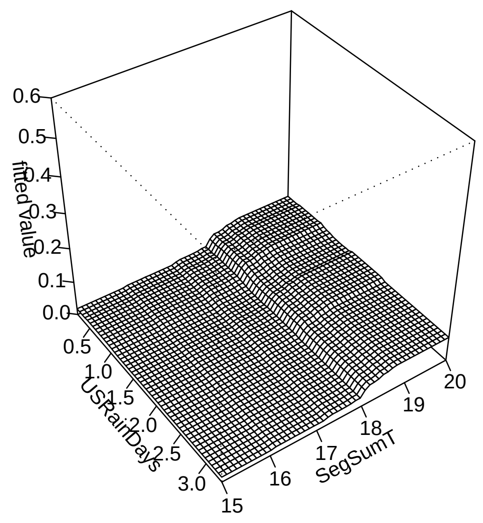
```
> find.int$rank.list
```

```
  var1.index var1.names var2.index var2.names int.size
1          9   USNative          1    SegSumT    23.38
2          6      USAvgT         1    SegSumT    10.71
3          9   USNative          5 DSMaxSlope     9.42
4         11     Method          9   USNative     8.42
5         11     Method          1    SegSumT     5.16
6          7 USRainDays          2   SegTSeas     4.97
```

You can plot pairwise interactions like this:

```
> gbm.perspec(angaus.tc5.lr005, 7, 1, y.range = c(15, 20), z.range = c(0,
+     0.6))
```

```
maximum value =  0.08
```



Additional options allow specifications of label names, rotations of the 3D graph and so on.

9 Predicting to new data

If you want to predict to a set of sites (rather than to a whole map), the general procedure is to set up a data.frame with rows for sites and columns for the variables that are in your model. R is case sensitive; the names need to exactly match those in the model. Other columns such as site IDs etc can also exist in the data.frame (and are ignored).

Our dataset for predicting to sites is in a file called Anguilla_test. The "Method" column needs to be converted to a factor, with levels matching those in the modelling data. To make predictions to sites from the BRT model use predict (or predict.gbm) from the gbm package The predictions are in a vector called preds. These are evaluation sites, and have observations in column 1 (named Angaus_obs). They are independent of the model building set and were used for an evaluation with independent data. Note that the calc.deviance function has different formulae for different distributions of data; the default is binomial, so we didn't specify it in the call

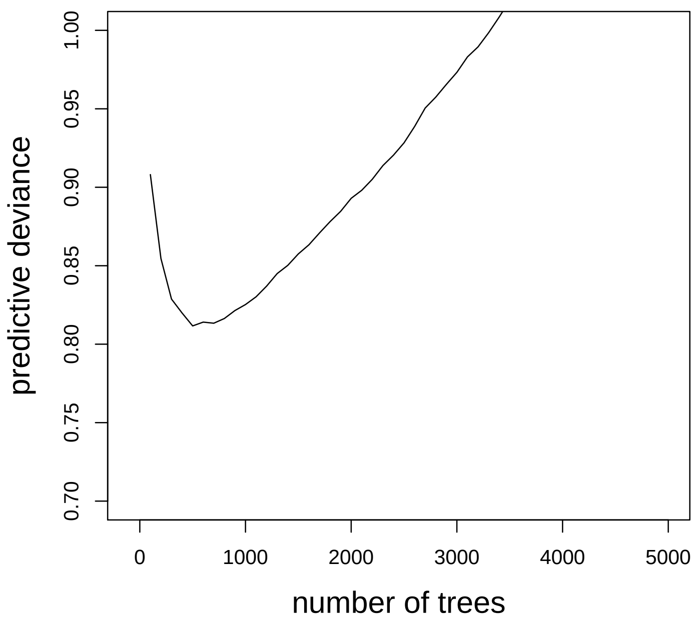
```
> data(Anguilla_test)
> preds <- predict.gbm(angaus.tc5.lr005, Anguilla_test, n.trees = angaus.tc5.lr005$gbm.call$
+     type = "response")
> calc.deviance(obs = Anguilla_test$Angaus_obs, pred = preds, calc.mean = TRUE)

[1] 0.8146787

> d = cbind(Anguilla_test$Angaus_obs, preds)
> pres = d[d[, 1] == 1, 2]
> abs = d[d[, 1] == 0, 2]
> e = evaluate(p = pres, a = abs)
> e

class            : ModelEvaluation
n presences      : 107
n absences       : 393
AUC              : 0.8234287
cor              : 0.4580961
TPR+TNR threshold: 0.117
```

One useful feature of prediction in gbm is you can predict to a varying number of trees. See the highlighted code below to how to predict to a vector of trees. The full set of code here shows how to make one of the graphed lines from Fig. 2 in our paper, using a model of 5000 trees developed with gbm.fixed

```
> angaus.5000 <- gbm.fixed(data = Anguilla_train, gbm.x = 3:13,
+     gbm.y = 2, learning.rate = 0.005, tree.complexity = 5, n.trees = 5000)

[1] fitting gbm model with a fixed number of 5000 trees for Angaus
[1] total deviance = 260.9
[1] residual deviance = 5.79

> tree.list <- seq(100, 5000, by = 100)
> pred <- predict.gbm(angaus.5000, Anguilla_test, n.trees = tree.list,
+     "response")
```

Note that the code above makes a matrix, with each column being the predictions from the model angaus.5000 to the number of trees specified by that element of tree.list - for example, the predictions in column 5 are for tree.list[5] = 500 trees. Now to calculate the deviance of all these results, and plot them:

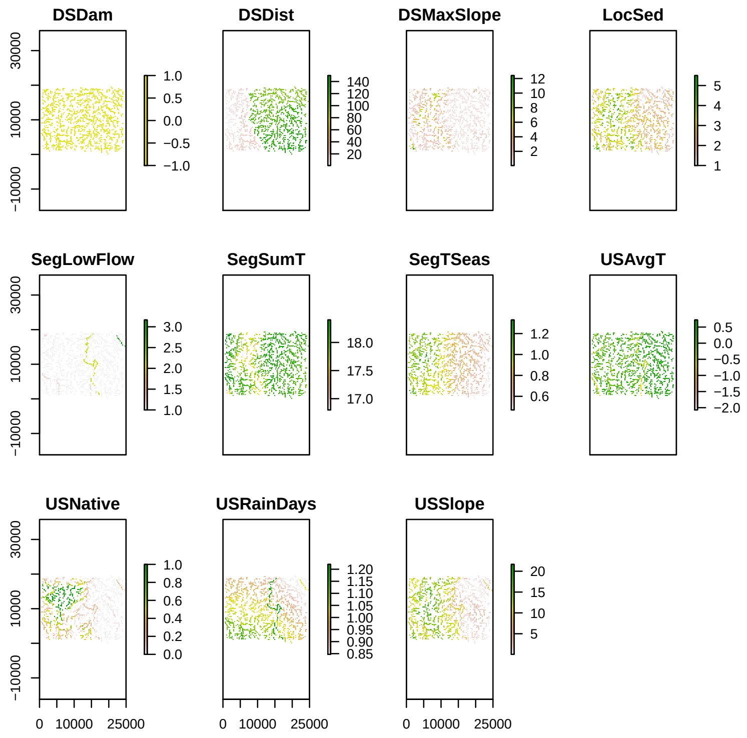
```
> angaus.pred.deviance <- rep(0, 50)
> for (i in 1:50) {
+     angaus.pred.deviance[i] <- calc.deviance(Anguilla_test$Angaus_obs,
+         pred[, i], calc.mean = TRUE)
+ }

> plot(tree.list, angaus.pred.deviance, ylim = c(0.7, 1), xlim = c(-100,
+     5000), type = "l", xlab = "number of trees", ylab = "predictive deviance",
+     cex.lab = 1.5)
```



10 spatial prediction

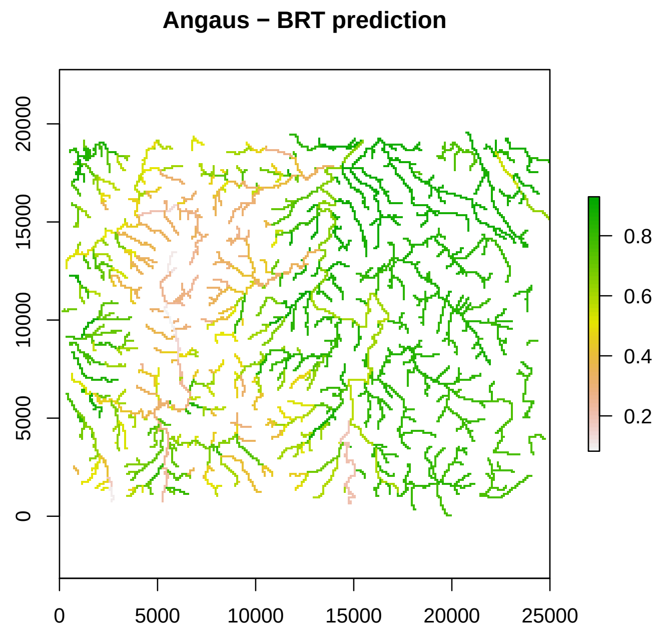
Here we show how to predict to a whole map (technically to a RasterLayer object) using the predict version in the package 'raster'. The predictor variables are available as a RasterBrick (multi-layered raster) in Anguilla_grids.

```
> data(Anguilla_grids)
> plot(Anguilla_grids)
```



There is (obviously) no grid for fishing method. We create a data.frame with a constant value (of class 'factor') and pass that on to the predict function.

```
> Method = factor("electric", levels = levels(Anguilla_train$Method))
> add = data.frame(Method)
> p = predict(Anguilla_grids, angaus.tc5.lr005, const = add, n.trees = angaus.tc5.lr005$gbm.
+     type = "response")
> p = mask(p, raster(Anguilla_grids, 1))
> plot(p, main = "Angaus - BRT prediction")
```



11 Further reading

The following includes a mix of both statistical papers and ecological applications:

Elith, J., Leathwick, J.R., and Hastie, T. (2008). Boosted regression trees - a new technique for modelling ecological data. Journal of Animal Ecology

Friedman, J.H. (2001) Greedy function approximation: a gradient boosting machine. The Annals of Statistics, 29, 1189-1232.

Friedman, J.H. (2002) Stochastic gradient boosting. Computational Statistics and Data Analysis, 38, 367-378.

Friedman, J.H., Hastie, T., and Tibshirani, R. (2000) Additive logistic regression: a statistical view of boosting. The Annals of Statistics, 28, 337-407.

Friedman, J.H. and Meulman, J.J. (2003) Multiple additive regression trees with application in epidemiology. Statistics in Medicine, 22, 1365-1381.

Hastie, T., R. Tibshirani, and J. H. Friedman. 2001. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, New York.

Leathwick, J.R., Elith, J., Francis, M.P., Hastie, T., and Taylor, P. (2006a) Variation in demersal fish species richness in the oceans surrounding New Zealand: an analysis using boosted regression trees. Marine Ecology Progress Series, 321, 267-281.

Moisen, G.G., Freeman, E.A., Blackard, J.A., Frescino, T.S., Zimmermann, N.E., and Edwards, T.C.. (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.

Ridgeway, G. (1999) The state of boosting. Computing Science and Statistics, 31, 172-181.

Ridgeway, G. (2006) Generalized boosted regression models. Documentation on the R package "gbm", version 1.5-7. http://www.i-pensieri.com/gregr/gbm.shtml.

Schapire, R. (2003). The boosting approach to machine learning - an overview. In MSRI Workshop on Nonlinear Estimation and Classification, 2002. (eds D.D. Denison, M.H. Hansen, C. Holmes, B. Mallick and B. Yu), Springer, NY.

Wintle, B. A., J. Elith, and J. Potts. 2005. Fauna habitat modelling and mapping in an urbanising environment; A case study in the Lower Hunter Central Coast region of NSW. Austral Ecology 30:729-748.