Boosted Regression Trees for ecological modeling

Jane Elith and John Leathwick

June 15, 2016

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)
> data(Anguilla_train)
> head(Anguilla_train)

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

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)

 GBM STEP - version 2.9

Performing cross-validation optimisation of a boosted regression tree model
```

```
for Angaus and using a family of bernoulli
Using 1000 observations and 11 predictors
creating 10 initial models of 50 trees

 folds are stratified by prevalence
total mean deviance =  1.0063
tolerance is fixed at  0.001
ntrees resid. dev.
50    0.8501
now adding trees...
100   0.7779
150   0.7401
200   0.7189
250   0.7067
300   0.6998
350   0.695
400   0.6906
450   0.6893
500   0.6887
550   0.6901
600   0.6892
650   0.6891
700   0.6886
750   0.6896
800   0.6894
850   0.6902
900   0.6912
950   0.6913
1000   0.6921
1050   0.6938
1100   0.6964
1150   0.6979
1200   0.6989

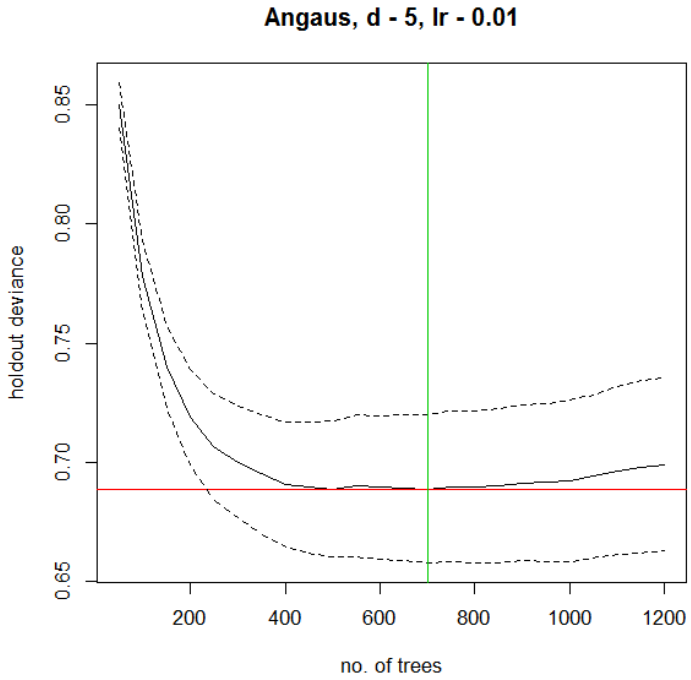
mean total deviance = 1.006
mean residual deviance = 0.438

estimated cv deviance = 0.689 ; se = 0.031

training data correlation = 0.795
cv correlation =  0.575 ; se = 0.03

training data AUC score = 0.964
cv AUC score = 0.871 ; se = 0.013

elapsed time -  0.21 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.

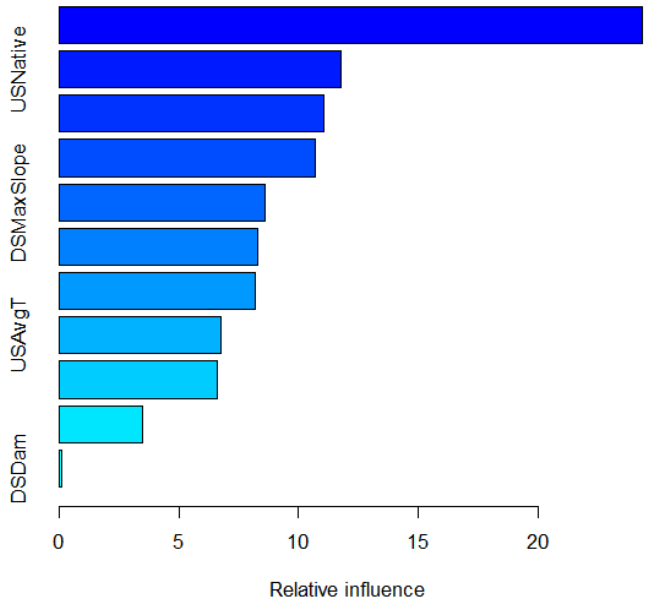
You can look at variable importance with the summary function

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

```
 [1] "initF"                "fit"
 [3] "train.error"          "valid.error"
 [5] "oobag.improve"        "trees"
 [7] "c.splits"             "bag.fraction"
 [9] "distribution"         "interaction.depth"
[11] "n.minobsinnode"       "num.classes"
[13] "n.trees"              "nTrain"
[15] "train.fraction"       "response.name"
[17] "shrinkage"            "var.levels"
[19] "var.monotone"         "var.names"
[21] "var.type"             "verbose"
[23] "data"                 "Terms"
[25] "cv.folds"             "call"
[27] "m"                    "gbm.call"
[29] "fitted"               "fitted.vars"
[31] "residuals"            "contributions"
[33] "self.statistics"      "cv.statistics"
[35] "weights"              "trees.fitted"
[37] "training.loss.values" "cv.values"
[39] "cv.loss.ses"          "cv.loss.matrix"
[41] "cv.roc.matrix"
```

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

```
                  var    rel.inf
SegSumT       SegSumT 24.3792376
USNative     USNative 11.8003350
Method         Method 11.0634091
DSDist         DSDist 10.7169075
DSMaxSlope DSMaxSlope  8.6052397
USSlope       USSlope  8.3120681
USRainDays USRainDays  8.1730486
USAvgT         USAvgT  6.7579857
SegTSeas     SegTSeas  6.5796152
SegLowFlow SegLowFlow  3.4989460
DSDam           DSDam  0.1132074
```



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 and using a family of bernoulli
Using 1000 observations and 11 predictors
creating 10 initial models of 50 trees

 folds are stratified by prevalence
total mean deviance =  1.0063
tolerance is fixed at  0.001
ntrees resid. dev.
50    0.908
```

```
now adding trees...
100   0.8458
150   0.804
200   0.7738
250   0.7525
300   0.7361
350   0.7245
400   0.7158
450   0.7092
500   0.7021
550   0.6974
600   0.6936
650   0.6899
700   0.687
750   0.6841
800   0.6821
850   0.68
900   0.6784
950   0.677
1000   0.676
1050   0.6748
1100   0.6741
1150   0.6734
1200   0.6728
1250   0.6713
1300   0.6703
1350   0.67
1400   0.6698
1450   0.6704
1500   0.6699
1550   0.6694
1600   0.6691
1650   0.6686
1700   0.6682
1750   0.6683
1800   0.6678
1850   0.6679
1900   0.6679
1950   0.6677
2000   0.668
2050   0.6686
2100   0.6693
2150   0.6697
2200   0.6697
2250   0.6702
```

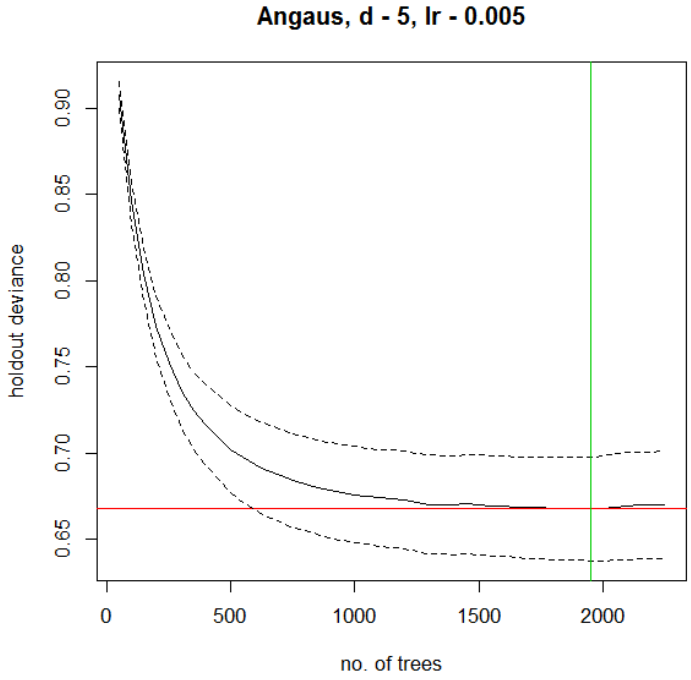
```
mean total deviance = 1.006
mean residual deviance = 0.379

estimated cv deviance = 0.668 ; se = 0.03

training data correlation = 0.837
cv correlation =  0.594 ; se = 0.025

training data AUC score = 0.977
cv AUC score = 0.878 ; se = 0.012

elapsed time -  0.41 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.

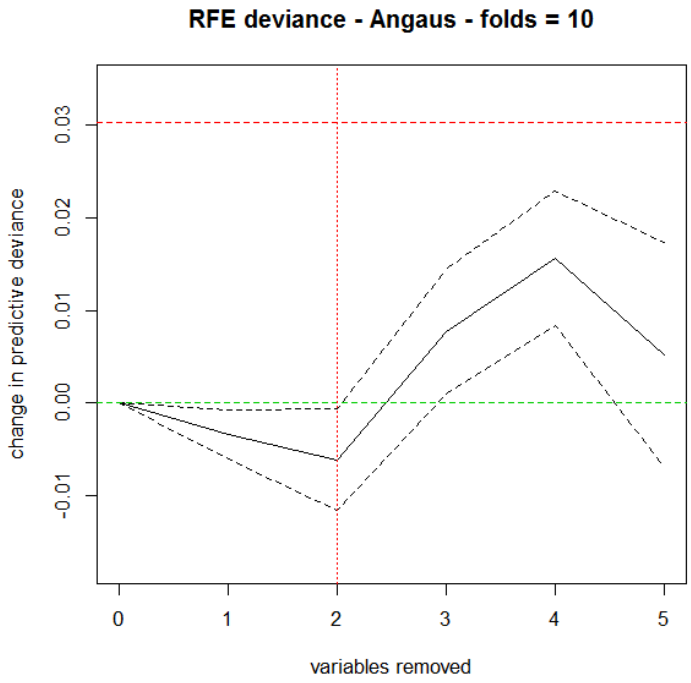
5 Alternative ways to fit models

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)
```



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 and using a family of bernoulli
Using 1000 observations and 10 predictors
creating 10 initial models of 50 trees

 folds are stratified by prevalence
total mean deviance =  1.0063
tolerance is fixed at  0.001
ntrees resid. dev.
50    0.9085
now adding trees...
100   0.8479
150   0.8065
200   0.7773
250   0.7553
300   0.7387
350   0.7266
400   0.7167
450   0.7081
500   0.702
550   0.6975
600   0.6944
650   0.6917
700   0.689
750   0.6867
800   0.6855
850   0.6849
900   0.6839
950   0.6834
1000   0.6823
1050   0.6819
```

```
1100   0.6819
1150   0.6813
1200   0.6809
1250   0.6803
1300   0.6802
1350   0.6792
1400   0.6794
1450   0.68
1500   0.6804
1550   0.6798
1600   0.6808
1650   0.681
1700   0.6811
1750   0.6808
1800   0.6809
1850   0.6813

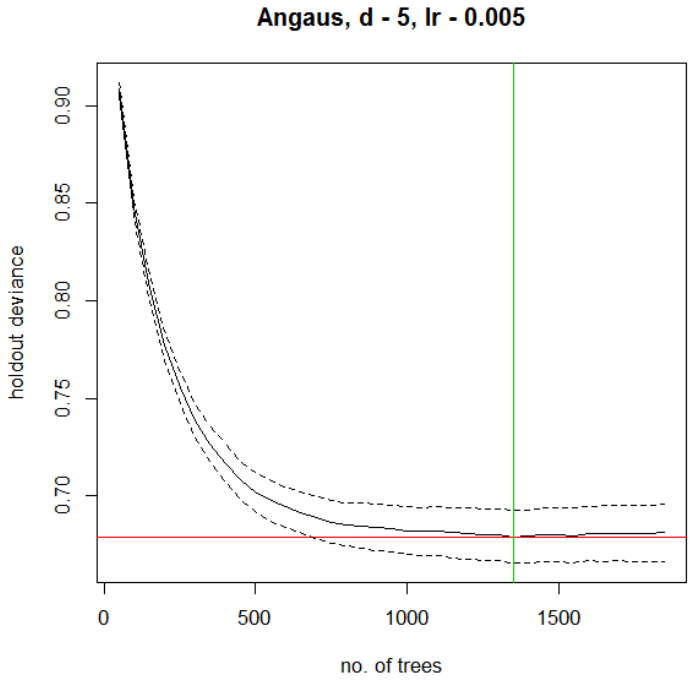
mean total deviance = 1.006
mean residual deviance = 0.438

estimated cv deviance = 0.679 ; se = 0.013

training data correlation = 0.8
cv correlation =  0.583 ; se = 0.015

training data AUC score = 0.966
cv AUC score = 0.876 ; se = 0.006

elapsed time -  0.31 minutes
```

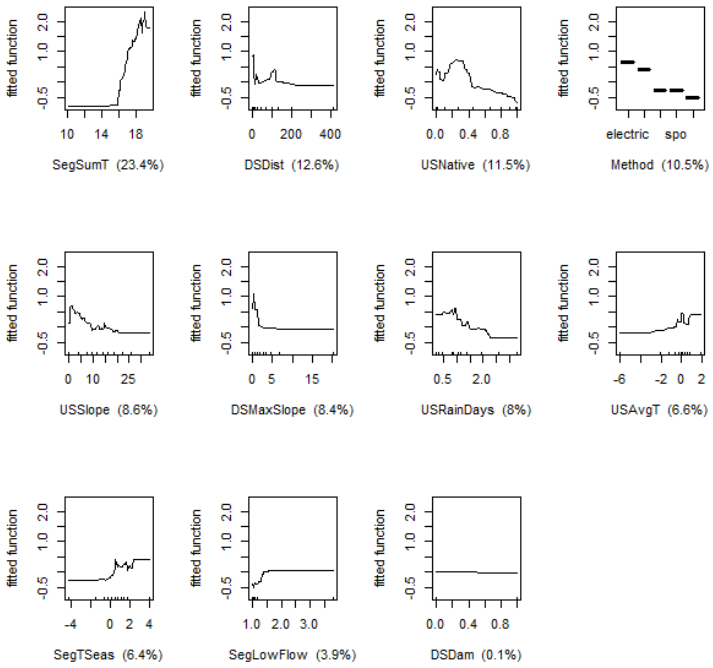


This has now made a new model (angaus.tc5.lr005.simp) with the same list components as described earlier. We could continue to use it, but given that we don't particularly want a more simple model (our view is that, in a dataset of this size, included variables that contribute little are acceptable),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

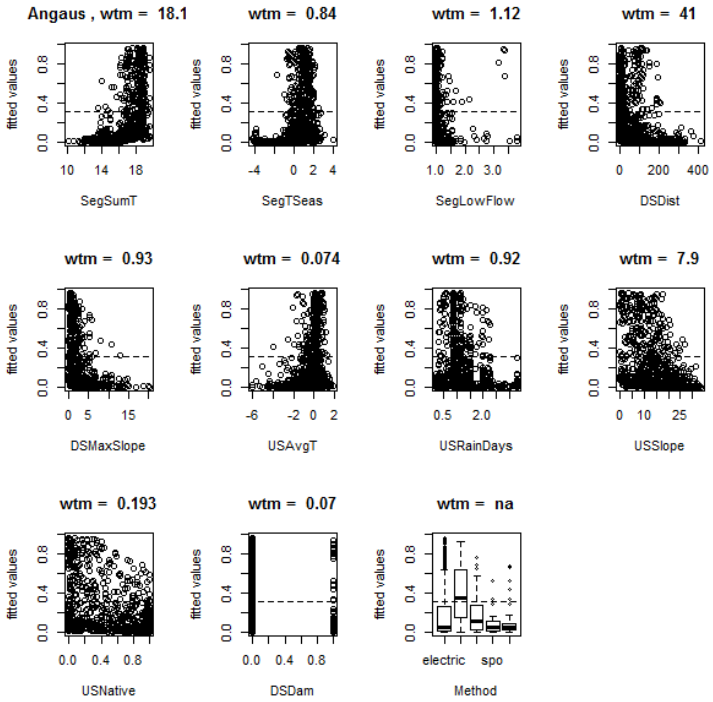
```
> gbm.plot(angaus.tc5.lr005, n.plots=11, write.title = FALSE)
```



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)
> find.int$interactions
```

	SegSumT	SegTSeas	SegLowFlow	DSDist	DSMaxSlope	USAvgT
SegSumT	0	3.71	0.11	24.16	2.04	8.52
SegTSeas	0	0.00	1.53	9.09	2.52	1.19
SegLowFlow	0	0.00	0.00	0.57	0.70	0.71
DSDist	0	0.00	0.00	0.00	0.10	0.14
DSMaxSlope	0	0.00	0.00	0.00	0.00	1.03

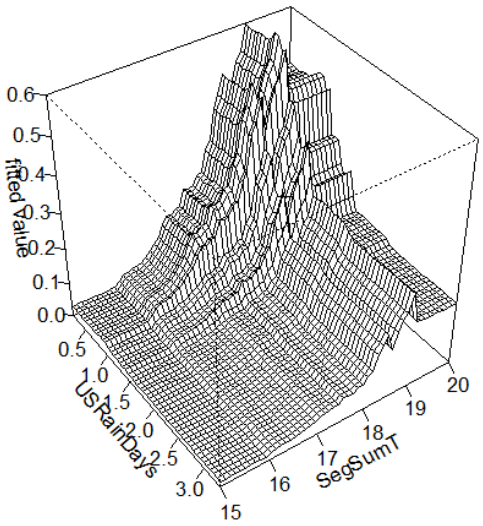
```
USAvgT           0     0.00       0.00   0.00       0.00   0.00
USRainDays       0     0.00       0.00   0.00       0.00   0.00
USSlope          0     0.00       0.00   0.00       0.00   0.00
USNative         0     0.00       0.00   0.00       0.00   0.00
DSDam            0     0.00       0.00   0.00       0.00   0.00
Method           0     0.00       0.00   0.00       0.00   0.00
           USRainDays USSlope USNative DSDam Method
SegSumT         27.06    5.02     8.89  0.01  13.97
SegTSeas         1.83    0.39     0.90  0.00   2.40
SegLowFlow       0.19    1.72     0.85  0.00   0.80
DSDist           7.10    2.17     1.69  0.00   2.76
DSMaxSlope       0.57    0.59     2.28  0.00   0.23
USAvgT           0.60    0.19     4.49  0.00   1.27
USRainDays       0.00    0.77     3.54  0.02   7.45
USSlope          0.00    0.00     2.15  0.00   5.43
USNative         0.00    0.00     0.00  0.01   1.70
DSDam            0.00    0.00     0.00  0.00   0.00
Method           0.00    0.00     0.00  0.00   0.00
```

```
> find.int$rank.list

  var1.index var1.names var2.index var2.names int.size
1          7 USRainDays          1    SegSumT    27.06
2          4     DSDist          1    SegSumT    24.16
3         11     Method          1    SegSumT    13.97
4          4     DSDist          2   SegTSeas     9.09
5          9   USNative          1    SegSumT     8.89
6          6     USAvgT          1    SegSumT     8.52
```

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))
```



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)
> library(gbm)
```

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

[1] 0.7454954

> 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            : 0.8592899
cor            : 0.5243171
max TPR+TNR at : 0.09215022
```

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)

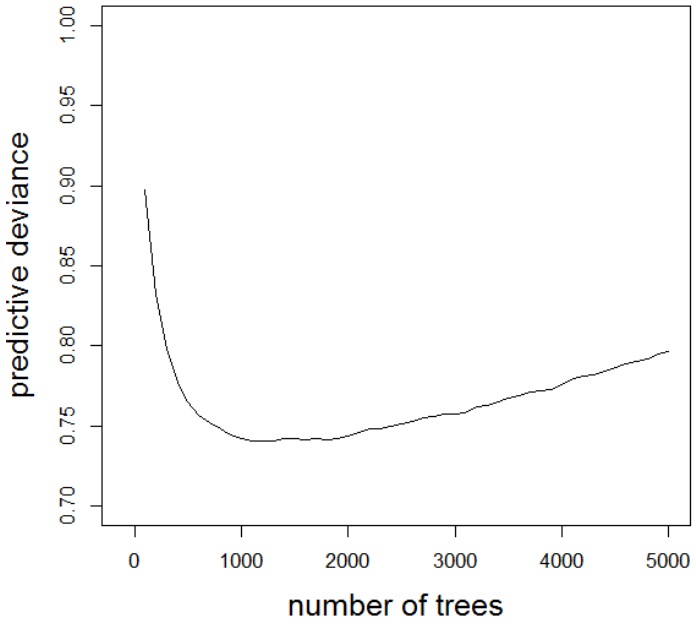
[1] fitting gbm model with a fixed number of 5000 trees for Angaus
[1] total deviance = 1006.33
[1] residual deviance = 203.81

> 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)
+ }

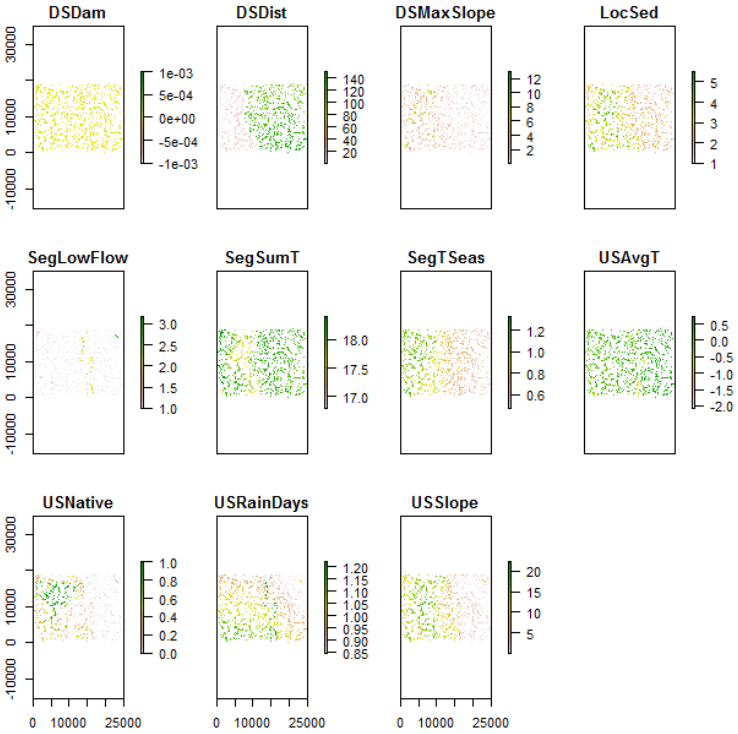
> plot(tree.list, angaus.pred.deviance, ylim=c(0.7,1), xlim=c(-100,5000),
+      type='l', xlab="number of trees", ylab="predictive deviance",
+      cex.lab=1.5)
```



10 Spatial prediction

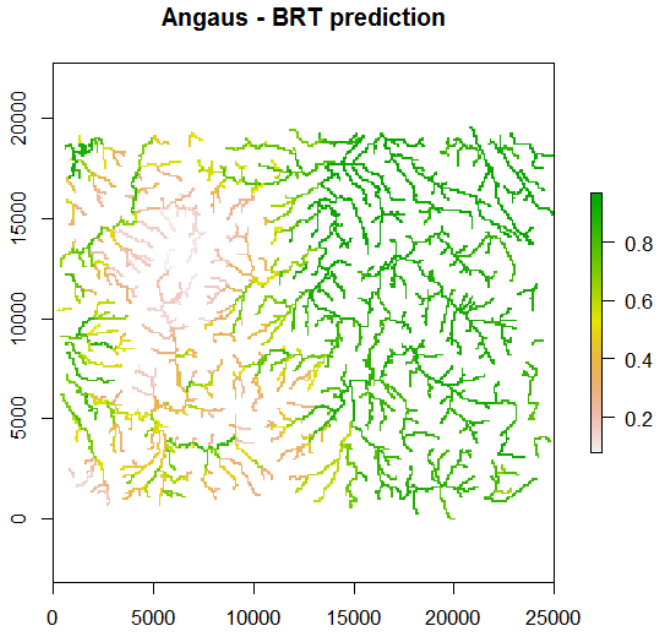
Here we show how to predict to a whole map (technically to a RasterLayer object) using the predict version in the package 'raster'. The predictor variables are available as a RasterBrick (multi-layered raster) in Anguilla_grids.

```
> 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.call$best.trees, 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.