Categorical soil attribute modeling and mapping: Random Forests.

Soil Security Laboratory

2018

1 Random Forests

The final model we will look at is the Random Forest, which we should be familiar with now as this model type was examined during the continuous variable prediction methods section. It can also be used for categorical variables. Some useful extractor functions like `print` and `importance` give some useful information about the model performance.

First lets prepare the data.

```
library(ithir)
library(sp)
library(raster)
library(rasterVis)

data(hvTerronDat)
data(hunterCovariates)
```

Transform the `hvTerronDat` data to a `SpatialPointsDataFrame`.

```
names(hvTerronDat)

## [1] "x"      "y"      "terron"

coordinates(hvTerronDat) <- ~x + y
```

As these data are of the same spatial projection as the `hunterCovariates`, there is no need to perform a coordinate transformation. So we can perform the intersection immediately.

```
DSM_data <- extract(hunterCovariates, hvTerronDat, sp = 1, method = "simple")
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)

## 'data.frame': 1000 obs. of  8 variables:
##  $ x               : num  346535 334760 340910 336460 344510 ...
##  $ y               : num  6371941 6375841 6377691 6382041 6378116 ...
##  $ terron          : Factor w/ 12 levels "1","2","3","4",..: 3 4 12 5 6 11 3 10 3 5 ..
##  $ AACN            : num  37.544 25.564 32.865 0.605 9.516 ...
##  $ Drainage.Index  : num  4.72 4.78 2 4.19 4.68 ...
```

```
##  $ Light.Insolation : num  1690 1736 1712 1712 1677 ...
##  $ TWI              : num  11.5 13.8 13.4 18.6 19.8 ...
##  $ Gamma.Total.Count: num  380 407 384 388 454 ...
```

It is always good practice to check to see if any of the observational data returned any NA values for any one of the covariates. If there is NA values, it indicates that the observational data is outside the extent of the covariate layers. It is best to remove these observations from the data set.

```
which(!complete.cases(DSM_data))
```

```
## integer(0)
```

```
DSM_data <- DSM_data[complete.cases(DSM_data), ]
```

Now we perform a random selection of data to hold back from the model fitting process.

```
set.seed(655)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
```

Now for the model fitting

```
library(randomForest)

hv.RF <- randomForest(terron ~ AACN + Drainage.Index + Light.Insolation + TWI +
    Gamma.Total.Count, data = DSM_data[training, ], ntree = 500, mtry = 5)

# Output random forest model diognostics
print(hv.RF)

# output relative importance of each covariate
importance(hv.RF)
```

Three types of prediction outputs can be generated from Random Forest models, and are specified via the `type` parameter of the `predict` extractor functions. The different “types” are the `response` (predicted class), `prob` (class probabilities) or `vote` (vote count, which really just appears to return the probabilities).

```
# Prediction of classes
predict(hv.RF, type = "response", newdata = DSM_data[training, ])

# Class probabilities
predict(hv.RF, type = "prob", newdata = DSM_data[training, ])
```

From the diagnostics output of the `hv.C5` model the confusion matrix is automatically generated, except it was a different orientation to what we have been looking for previous examples. This confusion matrix was performed on what is called the OOB or out-of-bag data i.e. it validates the model/s dynamically with observations withheld from the model fit. So lets just evaluate the model as we have done for the previous models. For calibration:

```r
C.pred.hv.RF <- predict(hv.RF, newdata = DSM_data[training, ])
goofcat(observed = DSM_data$terron[training], predicted = C.pred.hv.RF)
```

```
## $confusion_matrix
##      1 2  3  4  5  6   7   8  9 10 11 12
## 1   19 0  0  0  0  0   0   0  0  0  0  0
## 2    0 9  0  0  0  0   0   0  0  0  0  0
## 3    0 0 65  0  0  0   0   0  0  0  0  0
## 4    0 0  0 55  0  0   0   0  0  0  0  0
## 5    0 0  0  0 94  0   0   0  0  0  0  0
## 6    0 0  0  0  0 66   0   0  0  0  0  0
## 7    0 0  0  0  0  0 124   0  0  0  0  0
## 8    0 0  0  0  0  0   0 103  0  0  0  0
## 9    0 0  0  0  0  0   0   0 42  0  0  0
## 10   0 0  0  0  0  0   0   0  0 53  0  0
## 11   0 0  0  0  0  0   0   0  0  0 38  0
## 12   0 0  0  0  0  0   0   0  0  0  0 32
##
## $overall_accuracy
## [1] 100
##
## $producers_accuracy
##   1   2   3   4   5   6   7   8   9  10  11  12
## 100 100 100 100 100 100 100 100 100 100 100 100
##
## $users_accuracy
##   1   2   3   4   5   6   7   8   9  10  11  12
## 100 100 100 100 100 100 100 100 100 100 100 100
##
## $kappa
## [1] 1
```

It seems quite incredible that this particular model is indicating a 100% accuracy. Here it pays to look at the out-of-bag error of the `hv.RF` model for a better indication of the model goodness of fit. FOr the random holdback validation:

```r
V.pred.hv.RF <- predict(hv.RF, newdata = DSM_data[-training, ])
goofcat(observed = DSM_data$terron[-training], predicted = V.pred.hv.RF)
```

```
## $confusion_matrix
##     1 2  3  4  5  6  7  8 9 10 11 12
## 1   2 2  1  2  0  0  0  0 1  0  0  0
## 2   4 0  1  0  0  0  0  0 1  1  0  0
## 3   0 0 11  1  0  0  7  0 2  0  2  1
## 4   2 2  6 11  0  0  1  3 6  0  0  0
## 5   0 0  0  0 30  8  2  6 6  2  0  1
## 6   0 0  0  0  3 18  0  6 0  1  0  0
## 7   0 0 10  5  0  0 31  9 3  5  3  5
## 8   0 0  0  0  3  0  8 19 2  1  2  1
## 9   0 0  1  0  0  0  0  0 2  0  0  0
```

```
## 10 0 0  1  0  2  0  1  3 1 13  1  0
## 11 0 0  3  1  0  0  0  1 0  0  2  0
## 12 0 0  0  0  0  0  1  0 0  2  0  5
##
## $overall_accuracy
## [1] 48
##
## $producers_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 25  0 33 56 79 70 61 41  9 52 20 39
##
## $users_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 25  0 46 36 55 65 44 53 67 60 29 63
##
## $kappa
## [1] 0.4116538
```

So based on the model validation, the Random Forest performs quite similarly to the other models that were used before, despite a *perfect* performance based on the diagnostics of the calibration model.

And finally the map that results from applying the `hv.C5` model to the covariate rasters in shown on Figure 1 .

```
# class prediction
map.RF.c <- predict(covStack, hv.RF, filename = "hv_RF_class.tif", format = "GTiff",
    overwrite = T, datatype = "INT2S")

map.RF.c <- as.factor(map.RF.c)
rat <- levels(map.RF.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_002", "HVT_003", "HVT_004", "HVT_005",
    "HVT_006", "HVT_007", "HVT_008", "HVT_009", "HVT_010", "HVT_011", "HVT_012")
levels(map.RF.c) <- rat

# plot
area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#A8A800", "#0070FF",
    "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#B4D79E", "#FFFF00")
levelplot(map.RF.c, col.regions = area_colors, xlab = "", ylab = "")
```

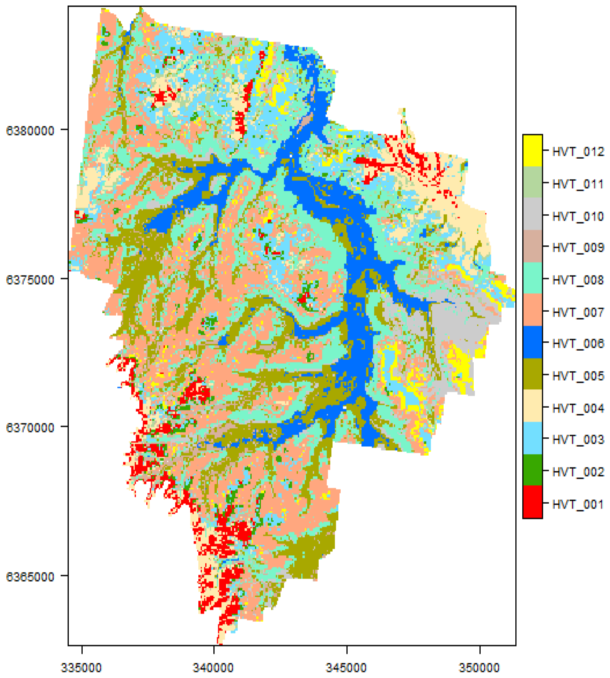

Figure 1: Hunter Valley Terron class map created using random forest model.