Continuous soil attribute modeling and mapping: Random Forest

Soil Security Laboratory

2018

1 Random Forests

An increasingly popular data mining algorithm in DSM and the soil sciences, and even in applied sciences in general is the Random Forests model. This algorithm is provided in the `randomForest` package and can be used for both regression and classification purposes. Random Forests are a boosted decision tree model. Further, Random Forests are an ensemble learning method for classification (and regression) that operate by constructing a multitude of decision trees at training time, which are later aggregated to give one single prediction for each observation in a data set.For regression the prediction is the average of the individual tree outputs, whereas in classification the trees vote by majority on the correct classification (mode). For further information regarding Random Forest and their underlying theory it is worth consulting Breiman (2001) and Grimm et al. (2008) as an example of its application in DSM studies.

Fitting a Random Forest model in `R` is relatively straightforward. It is worth consulting the the richly populated help files regarding the `randomForest` package and its functions. We will be using the `randomForest` function and a couple of extractor functions to tease out some of the model fitting diagnostics. Familiar will be the formula structure of the model. As for the Cubist model, there are many model fitting parameters to consider such as the number of trees to build (`ntree`) and the number of variables (covariates) that are randomly sampled as candidates at each decision tree split (`mtry`), plus many others. The `print` function allows one to quickly assess the model fit. The `importance` parameter (logical variable) used within the `randomForest` function specifies that we want to also assess the importance of the covariates used.

First get the data and perform the covariate data intersection

```
library(ithir)
library(raster)
library(rgdal)
library(sp)

# point data
```

```
data(HV_subsoilpH)

# Start afresh round pH data to 2 decimal places
HV_subsoilpH$pH60_100cm <- round(HV_subsoilpH$pH60_100cm, 2)

# remove already intersected data
HV_subsoilpH <- HV_subsoilpH[, 1:3]

# add an id column
HV_subsoilpH$id <- seq(1, nrow(HV_subsoilpH), by = 1)

# re-arrange order of columns
HV_subsoilpH <- HV_subsoilpH[, c(4, 1, 2, 3)]

# Change names of coordinate columns
names(HV_subsoilpH)[2:3] <- c("x", "y")


# grids (covariate raster)
data(hunterCovariates_sub)
```

Perform the covariate intersection.

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

# extract
DSM_data <- extract(hunterCovariates_sub, HV_subsoilpH, sp = 1, method = "simple")
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)
```

```
## 'data.frame': 506 obs. of  15 variables:
##  $ id                      : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ x                       : num  340386 340345 340559 340483 340734 ...
##  $ y                       : num  6368690 6368491 6369168 6368740 6368964 ...
##  $ pH60_100cm              : num  4.47 5.42 6.26 8.03 8.86 7.28 4.95 5.61 5.39 3.44 ...
##  $ Terrain_Ruggedness_Index: num  1.34 1.42 1.64 1.04 1.27 ...
##  $ AACN                    : num  1.619 0.281 2.301 1.74 3.114 ...
##  $ Landsat_Band1           : num  57 47 59 52 62 53 47 52 53 63 ...
##  $ Elevation               : num  103.1 103.7 99.9 101.9 99.8 ...
##  $ Hillshading             : num  1.849 1.428 0.934 1.517 1.652 ...
##  $ Light_insolation        : num  1689 1701 1722 1688 1735 ...
##  $ Mid_Slope_Positon       : num  0.876 0.914 0.844 0.848 0.833 ...
##  $ MRVBF                   : num  3.85 3.31 3.66 3.92 3.89 ...
##  $ NDVI                    : num  -0.143 -0.386 -0.197 -0.14 -0.15 ...
##  $ TWI                     : num  17.5 18.2 18.8 18 17.8 ...
##  $ Slope                   : num  1.79 1.42 1.01 1.49 1.83 ...
```

Often it is handy to check to see whether there are missing values both in the target variable and of the covariates. It is possible that a point location does not fit within the extent of the available covariates. In these cases the data should be excluded. A quick way to assess whether there are missing or NA

values in the data is to use the `complete.cases` function.

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

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

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

Now lets begin the Random Forest model fitting

```
library(randomForest)
set.seed(123)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))

# fit the model
hv.RF.Exp <- randomForest(pH60_100cm ~ AACN + Landsat_Band1 + Elevation + Hillshading +
    Mid_Slope_Positon + MRVBF + NDVI + TWI, data = DSM_data[training, ],
    importance = TRUE, ntree = 1000)

print(hv.RF.Exp)
```

```
##
## Call:
##  randomForest(formula = pH60_100cm ~ AACN + Landsat_Band1 + Elevation +
##       Hillshading + Mid_Slope_Positon + MRVBF + NDVI + TWI,
##         data = DSM_data[training,      ],
##             importance = TRUE, ntree = 1000)
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1.433532
##                     % Var explained: 21.1
```

Using the `varImpPlot` function allows one to visualize which covariates are of most importance to the prediction of our soil variable (Figure 1).

```
varImpPlot(hv.RF.Exp)
```

There is a lot of talk about how the variable importance is measured in Random Forest models. So it is probably best to quote from the source:

> *"Here are the definitions of the variable importance measures. For each tree, the prediction accuracy on the out-of-bag portion of the data is recorded. Then the same is done after permuting each predictor variable. The difference between the two accuracy measurements are then averaged over all trees, and normalized by the standard error. For regression, the MSE is computed on the out-of-bag data for each tree, and then the same computed after permuting a variable. The differences are averaged and normalized by the standard error. If the standard error is equal to 0 for a variable, the division is not done (but the measure is almost always equal to 0 in that case). For the node purity, it is the total decrease in node impurities from splitting on the variable, averaged over all*

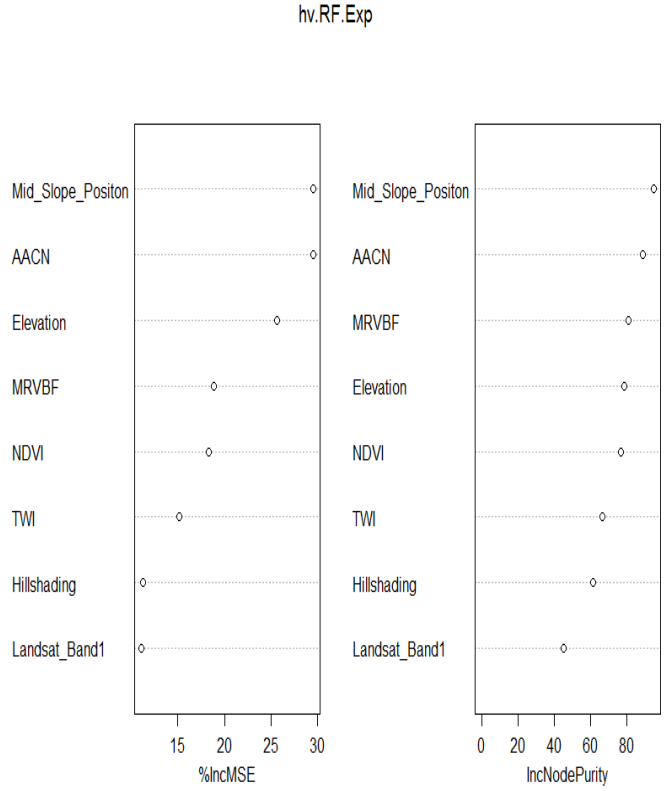

Figure 1: Covariate importance (ranking of predictors) from Random Forest model fitting.

> *trees. For classification, the node impurity is measured by the Gini index. For regression, it is measured by residual sum of squares."*

The important detail in the quotation above is the the variable importance is measured based on "out-of-bag" samples. Which means observation not included in the model. Another detail is that they are based on an MSE accuracy measure; in this case the difference, when a covariate is included and excluded in a tree model. The value is averaged over all trees.

So lets validate the `hv.RF.Exp` model both internally and externally.

```
# Internal validation
RF.pred.C <- predict(hv.RF.Exp, newdata = DSM_data[training, ])
goof(observed = DSM_data$pH60_100cm[training], predicted = RF.pred.C)
```

```
##          R2 concordance       MSE      RMSE        bias
## 1 0.9087286   0.8847666 0.2998826 0.5476153 0.003947595
```

```
# External validation
RF.pred.V <- predict(hv.RF.Exp, newdata = DSM_data[-training, ])
goof(observed = DSM_data$pH60_100cm[-training], predicted = RF.pred.V)
```

```
##           R2 concordance      MSE     RMSE       bias
## 1 0.1408058   0.2876967 1.534712 1.238835 0.04706017
```

Essentially these results are quite similar to those of the validations from the other models. One needs to be careful about accepting very good model fit results without a proper external validation. This is particularly poignant for Random Forest models which have a tendency to provide excellent calibration results which can often give the wrong impression about its suitability for a given application. Therefore when using random forest models it pays to look at the validation on the out-of-bag samples when evaluating the goodness of fit of the model.

So lets look at the map resulting from applying the `hv.RF.Exp` models to the covariates (Figure 2).

```
map.RF.r1 <- predict(hunterCovariates_sub, hv.RF.Exp, "soilpH_60_100_RF.tif",
    format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
plot(map.RF.r1, main = "Random Forest model predicted Hunter Valley soil pH (60-100cm)")
```

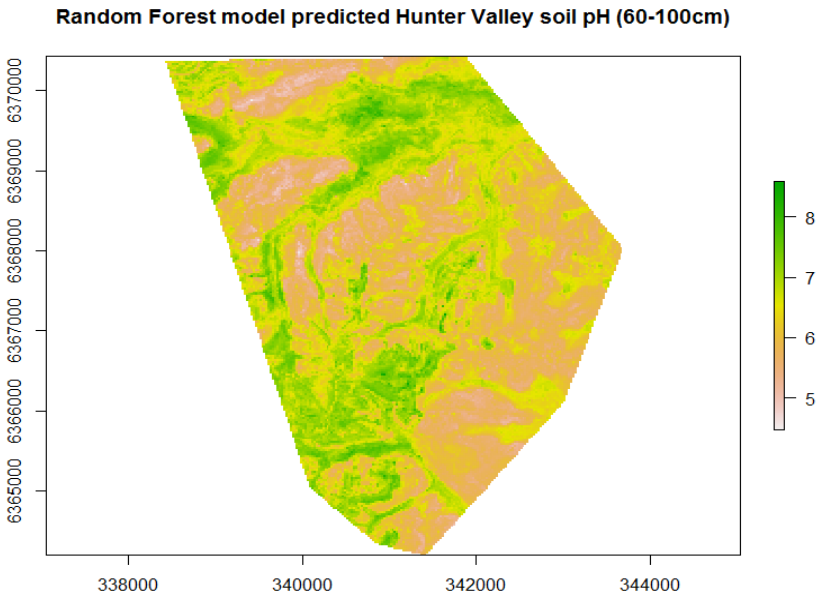

Figure 2: Random Forest model predicted Hunter Valley soil pH (60-100cm).

References

Breiman, L.
2001. Random forests. *Machine Learning*, 41:5–32.

Grimm, R., T. Behrens, M. Marker, and H. Elsenbeer

2008. Soil organic carbon concentrations and stocks on barro colorado island: Digital soil mapping using random forests analysis. *Geoderma*, 146:102–113.