# Bush density mapping

*CIAT*

*20 March, 2018, 11:57*

## 1 Objectives

This manual will help you conduct bush density mapping using climatic, vegetative and topographic data as covariates. At the end of this session, you will be able to:

1. Import data into R.
2. Create point shape file data for use in modelling.
3. Use the point and raster data to construct a random forest model.
4. Predict the random forest model.
5. Produce a bush density map

Before you start this session, it is important you have (i) the latest R software and (ii) Rstudio installed in your computer.

Note that this tutorial and code was prepared using R version 3.4.3 and packages from that version or later running on 64-bit OS, x64-based processor.

First clear your work space.

```
# clear your work space
rm(list = ls(all = TRUE))
```

Set the random seed. The number inside isn't important, you just need to ensure you use the same number each time so that the same random numbers are generated inside any random process in the session.

```
# set the random seed
set.seed(123)
```

## 2 Setting up your workspace

Let's set some variables e.g. input directories for use in the analysis.

```
# set variables
iDir <- "D:/OneDrive - CGIAR/ToBackup/_GitHub/LDN_Namibia/Bush_density_mapping/data"
oDir <- "D:/OneDrive - CGIAR/ToBackup/_GitHub/LDN_Namibia/Bush_density_mapping"
aoi <- "Otjiwarongo"
```

You need to first install all the required packages. Type (within R) install packages("name of package"), this needs to be done just once. You then load the packages using the 'library' function.

```
# load packages
library(sp)
library(rgdal)
library(raster)
library(randomForest)
library(plyr)
library(dplyr)
library(caret)
library(dygraphs)
```

```
library(car)
library(e1071)
library(snow)
library(plotKML)
```

The code below lists down the packages to be used in this session and installs if not already installed then loads into the current session.

```
# load packages
.packages = c("sp","rgdal","raster","randomForest","plyr",
              "dplyr","caret","car", "e1071","snow","plotKML")
.inst <- .packages %in% installed.packages()
if(length(.packages[!.inst]) > 0) install.packages(.packages[!.inst])
lapply(.packages, require, character.only=TRUE)
```

# 3 Importing data

Import point data into R. NB: The first row of the data file should contain the column names.

```
# read in data
raw.d <- read.csv(paste0(iDir, "/", aoi, "_bd_sampling_points", ".csv"), header=TRUE)
```

Sum all the shrubs per site for each category i.e. shrubs less 1.5m, shrubs more than 1.5m with no stem and shrubs more than 1.5m with stem.

```
# calculate values
raw.d$shrubs_less_1.5 <- apply(raw.d[,8:11], 1, sum, na.rm=TRUE)
raw.d$shrubs_more_1.5_no_stem <- apply(raw.d[,16:19], 1, sum, na.rm=TRUE)
raw.d$shrubs_more_1.5_stem <- apply(raw.d[,24:27], 1, sum, na.rm=TRUE)
```

Create new 'data.frame' with the few columns you will need to conduct this analysis. You will only need to type in the position of the column in the dataframe.

```
# create new dataframe with columns you need
raw.d<-raw.d[,c(1,2,3,34,35,36)]
```

Add a new column of shrubs more than 1.5m per site i.e. all subplots combined per site.

```
# add two new columns of shrubs > 1.5 and all shrubs in general
raw.d$shrubs_more_1.5 <- apply(raw.d[,5:6], 1, sum, na.rm=TRUE)
```

Subset the dataframe again to remove two not useful columns i.e. 'shrubs_more_1.5_no_stem' and 'shrubs_more_1.5_stem' and leave the columns you need.

```
# select the columns you need
raw.d<-raw.d[,c(1,2,3,4,7)]
```

The count of shrubs was done in sub plots with an area of 0.01ha, each site had four sub plots. To calculate shrubs per hectare we will therefore multiply the values by 25.

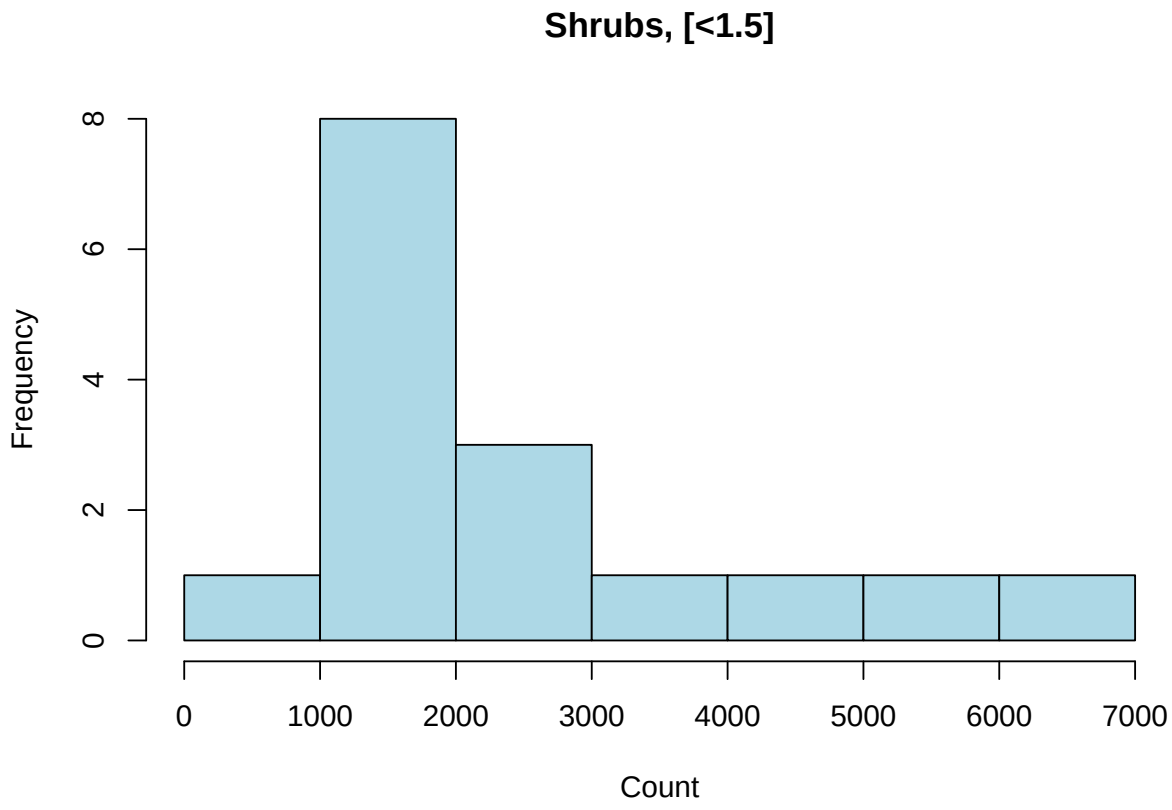
```
# compute shrubs per hectare
raw.d$shrubs_less_1.5 <- raw.d$shrubs_less_1.5*25
raw.d$shrubs_more_1.5 <- raw.d$shrubs_more_1.5*25
```

Remove all NAs.

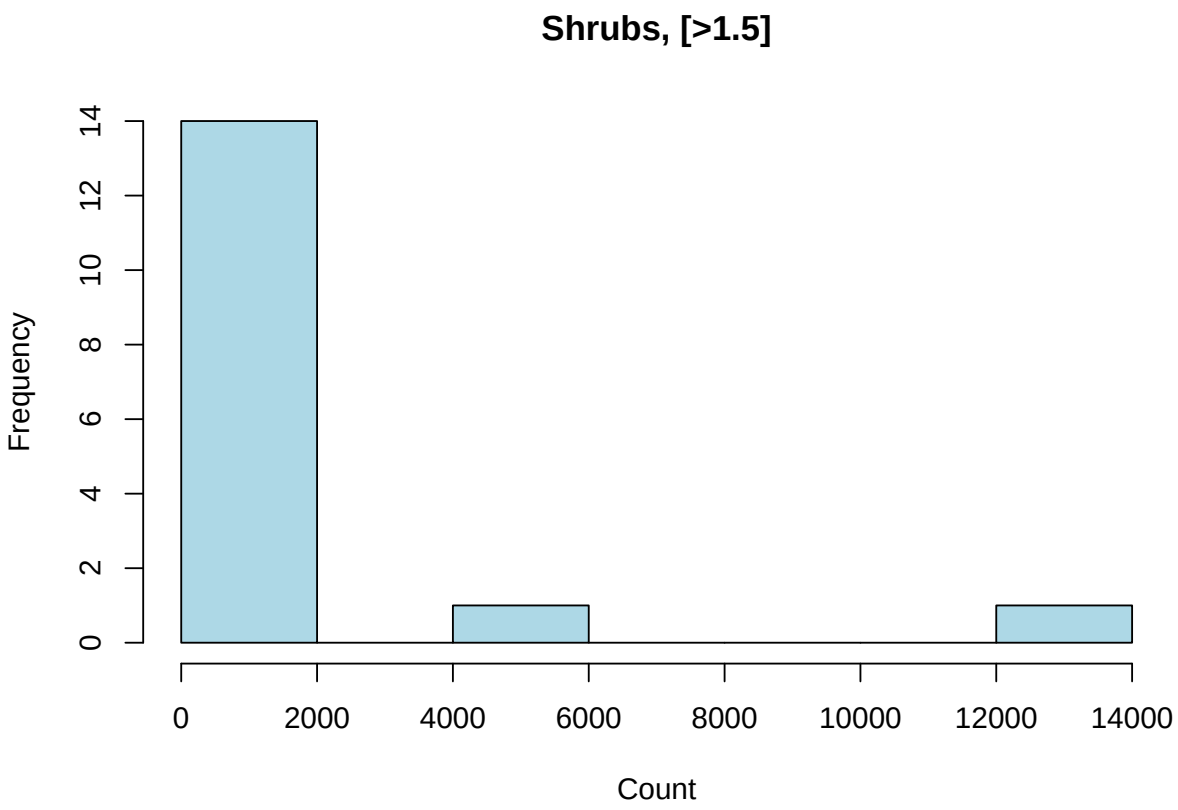
```
# remove all NAs
raw.d<-raw.d[complete.cases(raw.d),]
```

Plot histograms of the two variables

```
# plot histograms of the three variables
hist(raw.d$shrubs_less_1.5, col = "lightblue", xlab="Count", main="Shrubs, [<1.5]")
```




```
hist(raw.d$shrubs_more_1.5, col = "lightblue", xlab="Count", main="Shrubs, [>1.5]")
```




Create a separate folder to store outputs.

```
# create output folder
if (!dir.exists(paste0(oDir, "/", "outputs")))
  dir.create(paste0(oDir, "/", "outputs"))
```

Export the complete clean dataset to the 'output' folder just incase.

```
# export data to .csv
write.csv(raw.d, file = paste0(oDir, "/", "outputs", "/", aoi, "_OBS_Data", ".csv"),
          row.names=FALSE)
```

To convert the dataraframe into a spatial point dataframe, get the longitude and latitude from 'raw.d' object and always make sure that the order is in longitude/latitude.

```
# make shapefiles
xy <- raw.d[,c(3,2)]
trainDatageo <- SpatialPointsDataFrame(coords = xy, data = raw.d,
                                       proj4string = CRS("+proj=longlat
                                                         +datum=WGS84"))
trainData <- spTransform(trainDatageo, CRS('+proj=utm +zone=33 +south
                                           +datum=WGS84'))
```

Let's do some background checking of the field names and rename 'trainData' object fields.

```
# check column names
names(trainData)
```

```
## [1] "Waypoint_No"     "Latitude"        "Longitude"       "shrubs_less_1.5"
```

```
## [5] "shrubs_more_1.5"
```

Looks perfect, let us now import the rest of input data, stack and rename the contents.

```
# import the rest of input data and stack
r.list<-list.files(paste0(iDir, "/"), pattern = ".tif$", full.names = TRUE)
r.stack <- stack(r.list)
names(r.stack) <- c("asp","b1","b15","b16","b17","b18","cc","elev",
                    "l8b3","ndvi","soil")
```

Note that we are only calculating bush density in the bush area LULC catageory, therefore we need to mask out areas that are not bush.

```
# import the bush area mask
o.mask <- raster(paste0(iDir, "/", "other_data", "/", "Otji_BushArea_2016", ".tif"))
```

Set extent of the training data to match that of covariates.

```
# set extent of the training data
trainData@bbox <- bbox(o.mask)
```

Mask, read and stack the covariates. You will need to do some clean up by removing all NA values otherwise they will bring problems in the model.

```
# mask and remove NAs in the covariates
covs <- mask(r.stack, o.mask )
covs <- na.omit(covs)
```

Extract the cell values from the covariates based on the 'trainData' and attach the values in the attribute table of 'trainData' object.

```
# assign raster values to the training data
v<-as.data.frame(extract(covs,trainData))
trainData@data=data.frame(trainData@data, v[match(rownames(trainData@data),
                                                  rownames(v)),])
```

Rename fields in the training dataset, remove NAs and write the dataset as a .csv file for use later.

```
# rename fields in the training dataset, remove NAs, write the dataset
names(trainData) <- c("waypoint.no","lat","lon","shrubs.l1.5","shrubs.g1.5",
                      "asp","b1","b15","b16","b17","b18","cc", "elev","l8b3",
                      "ndvi","soil")
trainData@data<-trainData@data[complete.cases(trainData@data),]
write.csv(trainData@data, file = paste0(oDir, "/", "outputs", "/", aoi, "_trainingData.csv"), row.names=
```

# 4 Data exploration

Let's look at the distribution counts. We calculate the skewness statistic using the 'skewness' function from 'e1071' package.

```
# convert trainData object to data.frame
d <- as.data.frame(trainData@data, na.rm=TRUE)

# skewness statistics for shrubs <1.5m
skewness(d$shrubs.l1.5, na.rm = T)
```

```
## [1] 0.694236
```

```r
# skewness statistics for shrubs >1.5m
skewness(d$shrubs.g1.5, na.rm = T)
```

```
## [1] 1.979185
```

Compute correlation coefficients and plot correlations.

```r
# compute correlation coefficients and plot correlations
cor(d[,4:16])
pairs(d[,4:16])
```

Correlate count of shrubs with NDVI and Landsat 8 band 2-7.

```r
# correlate count of shrubs with NDVI and Landsat 8 band 2-7
cor(d$shrubs.l1.5,d$cc)
cor(d$shrubs.g1.5,d$cc)
```

This dataset is skewed as such we need to transform it.

```r
# histogram before transformation
hist(d$shrubs.l1.5)
plot(shrubs.l1.5~b1,data=d)

# histogram after transformation
hist(log(d$shrubs.l1.5))
plot(log(shrubs.l1.5)~log(b1),data=d)

# regression model for log tranformed data for shrubs<1.5
reg1 <- lm(log(shrubs.l1.5)~log(b1),data=d)
summary(reg1)

# regression model for log tranformed data for shrubs>1.5
reg2 <- lm(log(shrubs.g1.5)~log(b1),data=d)
summary(reg2)
```

We can also test for sigficance and see which independent variables in the multiple linear regression model of the dataset "d" are statistically significant at .05 significance level.

```r
# log tranformed lm1
x1 <- log(shrubs.l1.5)~b1+b15+cc+ndvi+soil
lm1 <- lm(x1,data=d)

# log tranformed lm2
x2 <- log(shrubs.g1.5)~b1+b15+cc+ndvi+soil
lm2 <- lm(x2,data=d)
```

```r
# model 1 summary
summary(lm1)

# model 2 summary
summary(lm2)
```

The R-squared of 'lm1' and 'lm2' is 0.722643 and 0.8929241 respectively, which means 72.2643034 and 89.2924063 of the variance in the dependent variables can be explained by the set of predictors in the two models.

At this point we can just fit a linear regression model but it only works nicely when data has a linear shape. In our case, the data has a non-linear shape thus the linear model cannot capture the non-linear features.

# 5 Model fitting

Specify the model inputs i.e. response variables and covariate values. Note that we have two response variables i.e. shrubs <1.5m and shrubs >1.5m. We will store these objects separately.

```
# select covariate values
s.cov <- d[,c(6:16)]

# select response variable [shrubs.l1.5]
s.l1.5<-d[,4]

# select response variable [shrubs.g1.5]
s.g1.5<-d[,5]
```

For more information about fitting a 'randomForest' model see the randomForest package. You can now fit the two models specifying the models as a formulas with the dependent variable i.e., count of shrubs.

```
# fit random forest model for shrubs <1.5m
model1 <- randomForest(x = s.cov, y = s.l1.5, ntree=5000,
                       mtry=5, importance=T)

# fit random forest model for shrubs >1.5m
model2 <- randomForest(x = s.cov, y = s.g1.5, ntree=5000,
                       mtry=5, importance=T)

# predicted versus observed
cor(model1$predicted,model1$y)**2
```

```
## [1] 0.5198277
```

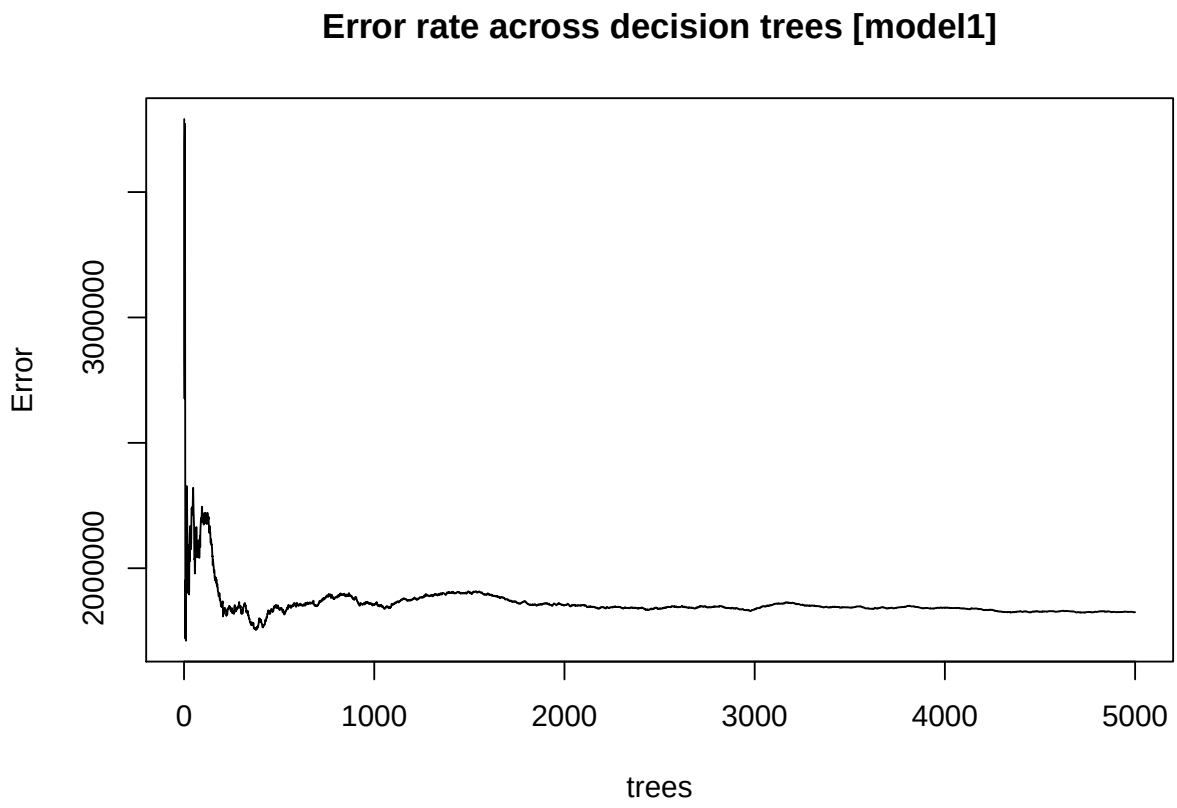
```
cor(model2$predicted,model2$y)**2
```

```
## [1] 0.3558012
```

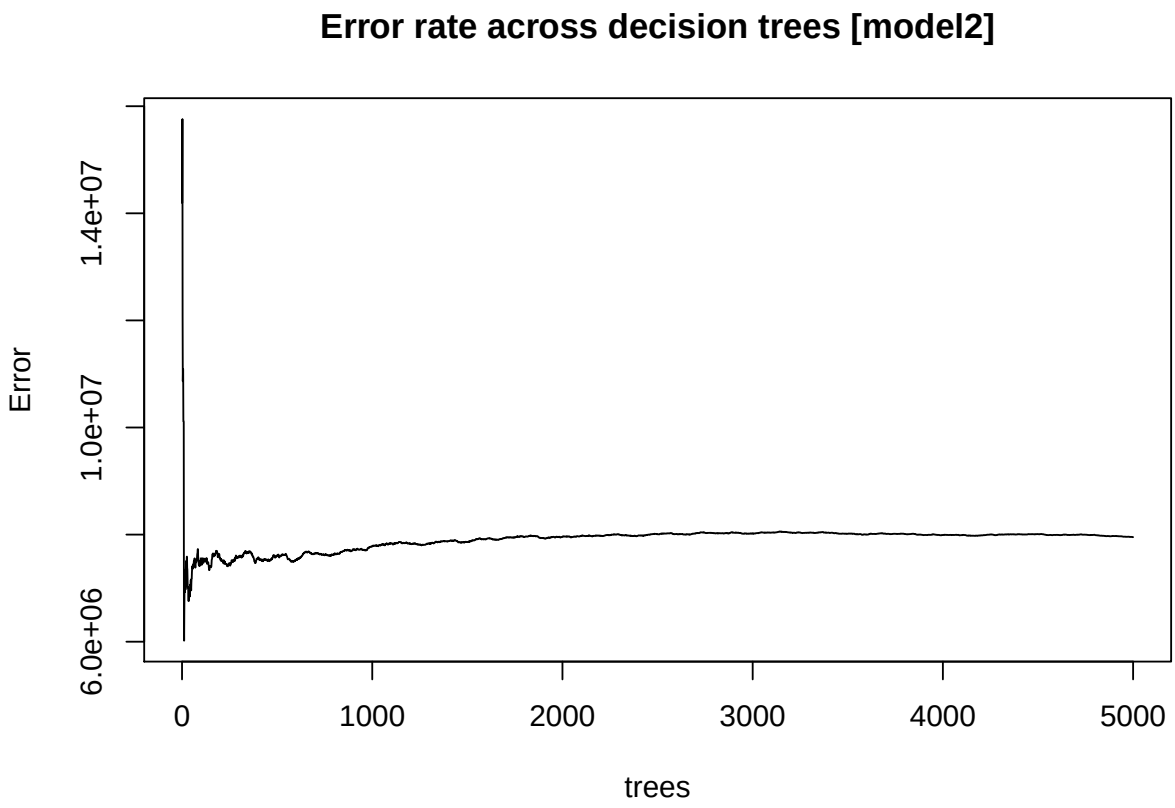
```
# get trees
getTree(model1,k=2)
getTree(model2,k=2)

# explore the models
str(model1, max.level=2)
str(model2, max.level=2)
```

Let us plot the two models and look at the error rate across decision trees. At what point is there no significant reduction in error rate?

```
# plot model 1
plot(model1, main = "Error rate across decision trees [model1]")
```

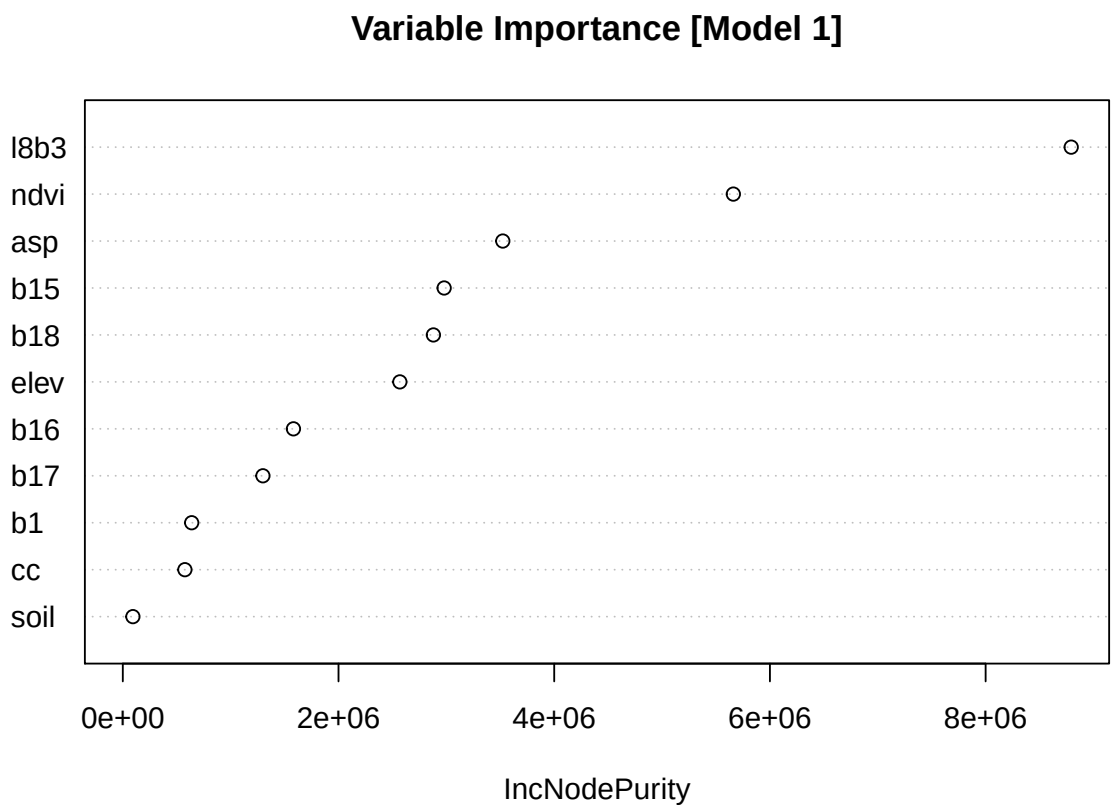



```
# plot model 2
plot(model2, main = "Error rate across decision trees [model2]")
```

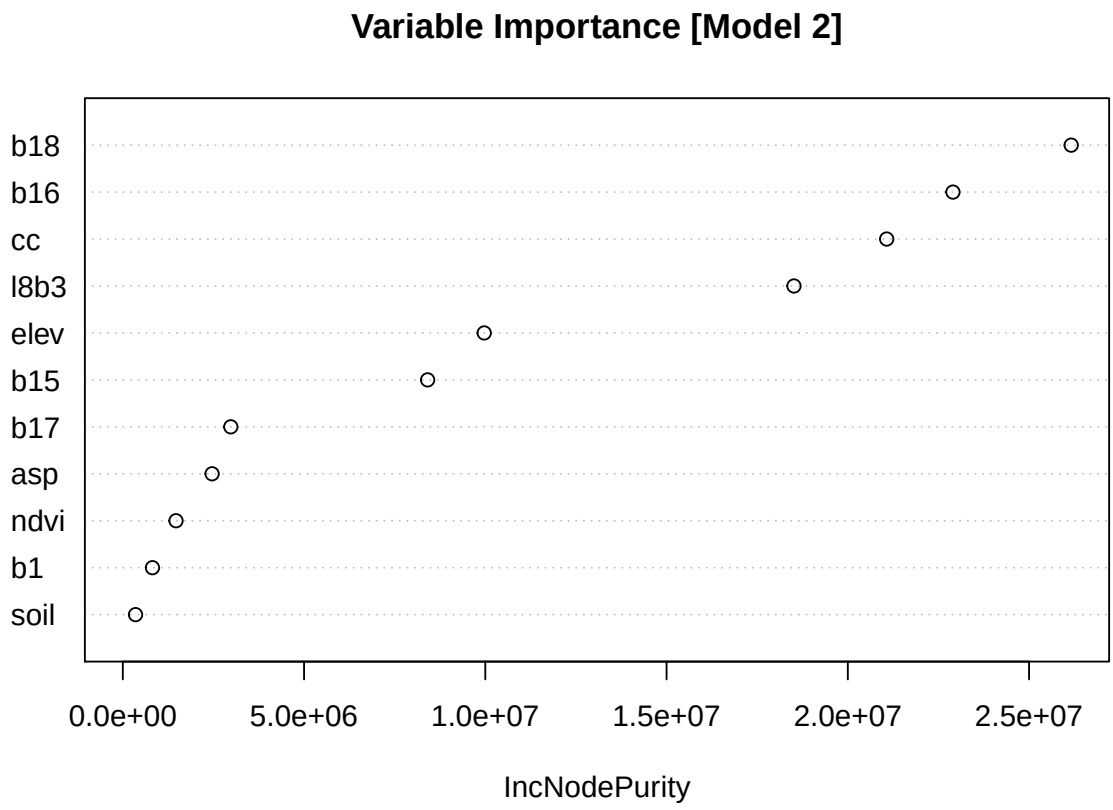



Another very useful tool is 'Variable importance' plot, this is achieved using the `varImpPlot` function and is based on the Model Accuracy and Gini value. The output is a table with decreasing order of importance based on a measure i.e. 1 = model accuracy and 2 = node impurity.

```
# variable importance plot for model 1
varImpPlot(model1, type = 2, main="Variable Importance [Model 1]")
```




```
# variable importance plot for model 2
varImpPlot(model2, type = 2, main="Variable Importance [Model 2]")
```




We have plotted 'type 2' variable improtance above. Node purity is a measure of how homogeneous a node is. A high score means the variable was important.

# 6 Model prediction

Use the 'predict' command to make rasters with predictions from the fitted models. To speed up computations use the 'clusterR' function from the 'raster' package which supports multi-core computing for functions such as predict (NB: install 'snow' package).

```
# predict models
beginCluster()
```

```
## 4 cores detected, using 3
```

```
prediction1 <- clusterR(covs, raster::predict, args = list(model = model1))
prediction2 <- clusterR(covs, raster::predict, args = list(model = model2))
endCluster()
```

Round raster values to whole numbers and save the predicted images as GeoTIFFs.

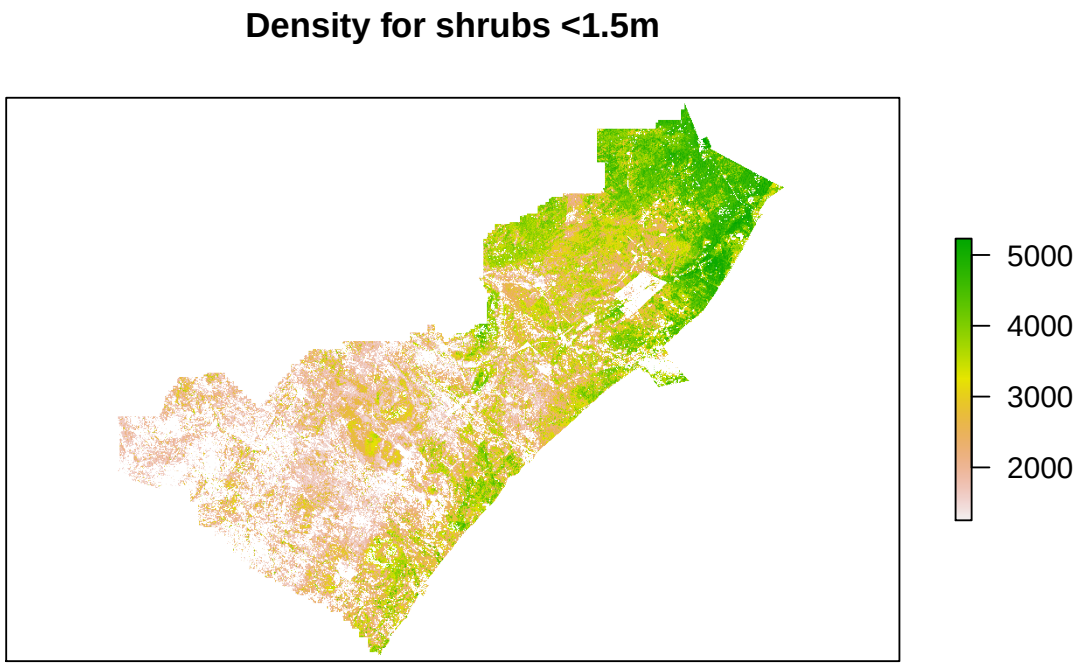
```
# round the numbers
prediction1<-round(prediction1, digits = 0)
prediction2<-round(prediction2, digits = 0)
writeRaster(prediction1, filename = paste0(oDir, "/",
                                           "outputs", "/", aoi, "_bd1",
                                           ".tif"), overwrite=TRUE)
```

```
writeRaster(prediction2, filename = paste0(oDir, "/",
                                           "outputs", "/", aoi, "_bd2",
                                           ".tif"), overwrite=TRUE)
```

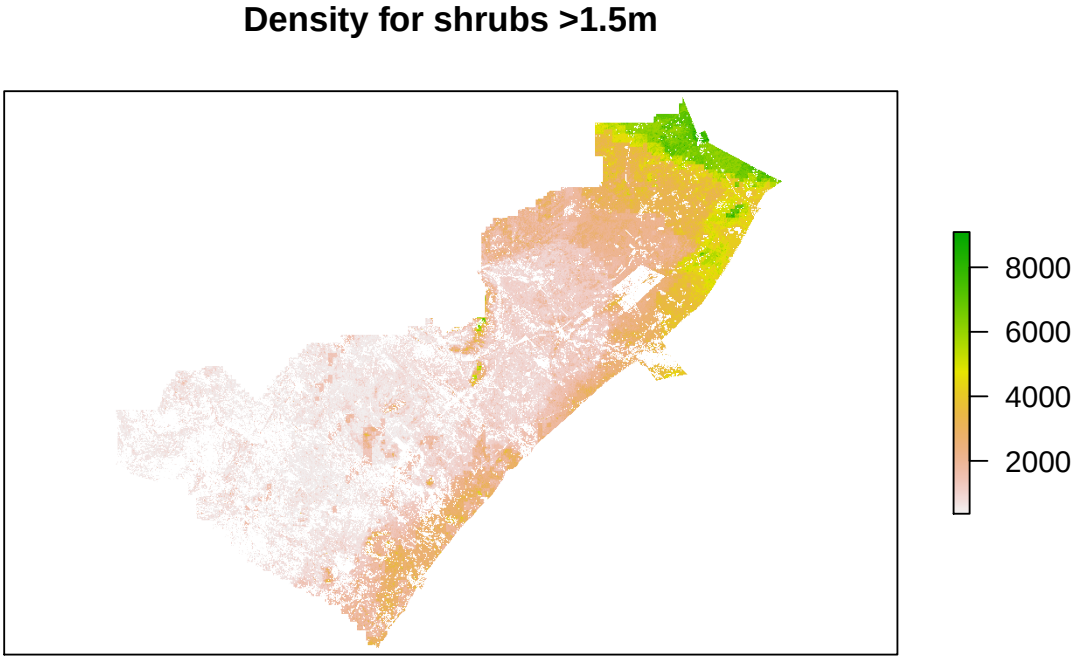
# 7 Plot results

Plot the two maps.

```
# plot the two maps
plot(prediction1, main="Density for shrubs <1.5m", axes=FALSE)
```




```
plot(prediction2, main="Density for shrubs >1.5m", axes=FALSE)
```


Density for shrubs >1.5m
8000
6000
4000
2000