Bush density mapping

CIAT

11 October, 2016, 23:36

Objectives

This manual will help you calculate bush density using count data with ndvi and crown cover data as covariates. At the end of this session, you will be able to:

1. Import excel data into R
2. Import raster data into R
3. Convert data from excel to ESRI point shapefile
4. Use the point and raster data to construct a random forest model (RF)
5. Use the RF model to predict bush density in areas we don't have count data
6. Test how accurate the model is

Download the sample dataset we will use in this session from this link https://drive.google.com/open?id=0B_Gkb_0tNKkQcGZGc1ZWRnpZc0U

You can now start the session but first clear your work space:

```
rm(list = ls(all = TRUE))
```

To enable us reproduce the results next time, let's set the seed

```
set.seed(500)
```

Let's set the start of data processing

```
startTime <- Sys.time()
cat("Start time", format(startTime),"\n")
```

```
## Start time 2016-10-11 23:36:45
```

Set up some options (you do not have to do this)

Set working directory

```
setwd("C:/LDN_Workshop/Sample_dataset/Bush_Density_Mapping")
```

Let's list down the packages to be used in this session. Packages will be installed if not already installed. They will then be loaded into the session

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

```
## Loading required package: sp
```

```
## Loading required package: rgdal
```

```
## rgdal: version: 1.1-8, (SVN revision 616)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.0.1, released 2015/09/15
##  Path to GDAL shared files: C:/Users/jymutua/Documents/R/win-library/3.1/rgdal/gdal
##  GDAL does not use iconv for recoding strings.
##  Loaded PROJ.4 runtime: Rel. 4.9.1, 04 March 2015, [PJ_VERSION: 491]
##  Path to PROJ.4 shared files: C:/Users/jymutua/Documents/R/win-library/3.1/rgdal/proj
##  Linking to sp version: 1.2-3
```

```
## Loading required package: raster
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## Loading required package: plyr
```

```
## Loading required package: xlsx
```

```
## Loading required package: rJava
```

```
## Loading required package: xlsxjars
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:raster':
##
##     intersect, select, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: car

## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
```

```
##
## [[6]]
## [1] TRUE
##
## [[7]]
## [1] TRUE
##
## [[8]]
## [1] TRUE
##
## [[9]]
## [1] TRUE
##
## [[10]]
## [1] TRUE
```

To get help on the functions and data sets in R, use `help()` or `?`. For example, to view the help file for the `calc` function, type one of the following:

```
help(calc)
?calc
```

Read in data from excel sheet

```
BushData <- read.xlsx("Field_data/Otji_BD_Sampling_Points.xlsx", sheetName =
                        "Sheet1", header=TRUE)
```

Calculate values by finding the median in crown cover and mean of values for counts.Note that 1 plot = 0.01 ha; 4 plots = 0.04 ha

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

Create new data frame with the columns you need

```
BushData_clean<-BushData[,c("Waypoint_No","Latitude","Longitude",
                            "shrubs_less_1.5","shrubs_more_1.5_no_stem",
                            "shrubs_more_1.5_stem")]
```

Add two new columns of shrubs more than 1.5 and all shrubs in general

```
BushData_clean$shrubs_more_1.5 <- apply(BushData_clean[,5:6], 1, sum,
                                        na.rm=TRUE)
BushData_clean$shrubs_all <- apply(BushData_clean[,4:6], 1, sum, na.rm=TRUE)
```

Round columns

```
BushData_clean<-BushData_clean %>%
  mutate_each(funs(round(.,0)), shrubs_less_1.5, shrubs_more_1.5_no_stem,
              shrubs_more_1.5_stem, shrubs_more_1.5, shrubs_all)
```

Remove all NAs

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

Export data to .csv

```
write.csv(BushData_clean, file = "Otji_BushData_trainData.csv",row.names=FALSE)
```

Get long and lat from your data.frame. Make sure that the order is in lon/lat. Convert the dataraframe into a spatial point dataframe

```
xy <- BushData_clean[,c(3,2)]
trainDatageo <- SpatialPointsDataFrame(coords = xy, data = BushData_clean,
                                       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

```
names(trainData)
```

```
## [1] "Waypoint_No"             "Latitude"
## [3] "Longitude"               "shrubs_less_1.5"
## [5] "shrubs_more_1.5_no_stem" "shrubs_more_1.5_stem"
## [7] "shrubs_more_1.5"         "shrubs_all"
```

rename trainData fields

```
names(trainData) <- c("Waypoint_No","Latitude","Longitude","shrubs_less_1.5",
                      "shrubs_more_1.5_no_stem", "shrubs_more_1.5_stem",
                      "shrubs_more_1.5", "shrubs_all")
```

Import the rest of input data

```
rasList <- list.files("Raster_data/", pattern = ".tif$", full.names = TRUE)
```

Create a raster stack and rename the contents

```
rstack <- stack(rasList)
names(rstack) <- c("crown_cover","NDVI","band2","band3","band4","band5",
                   "band6","band7")
```

Note that we are only calculating bush density in the bush area LULC catageory. Import the bush area mask

```
Otjo_BushArea <- raster("Raster_data/Other_data/Otji_BushArea_2016.tif")
```

Let's check various stuff

```
crs(rstack)
```

```
## CRS arguments:
##  +proj=utm +zone=33 +south +datum=WGS84 +units=m +no_defs
## +ellps=WGS84 +towgs84=0,0,0
```

```
crs(Otjo_BushArea)
```

```
## CRS arguments:
##  +proj=utm +zone=33 +south +datum=WGS84 +units=m +no_defs
## +ellps=WGS84 +towgs84=0,0,0
```

```
crs(trainData)
```

```
## CRS arguments:
##  +proj=utm +zone=33 +south +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
```

```
extent(rstack)
```

```
## class       : Extent
## xmin        : 559815
## xmax        : 701205
## ymin        : 7674145
## ymax        : 7792225
```

```
extent(Otjo_BushArea)
```

```
## class       : Extent
## xmin        : 559815
## xmax        : 701205
## ymin        : 7674145
## ymax        : 7792225
```

```
extent(trainData)
```

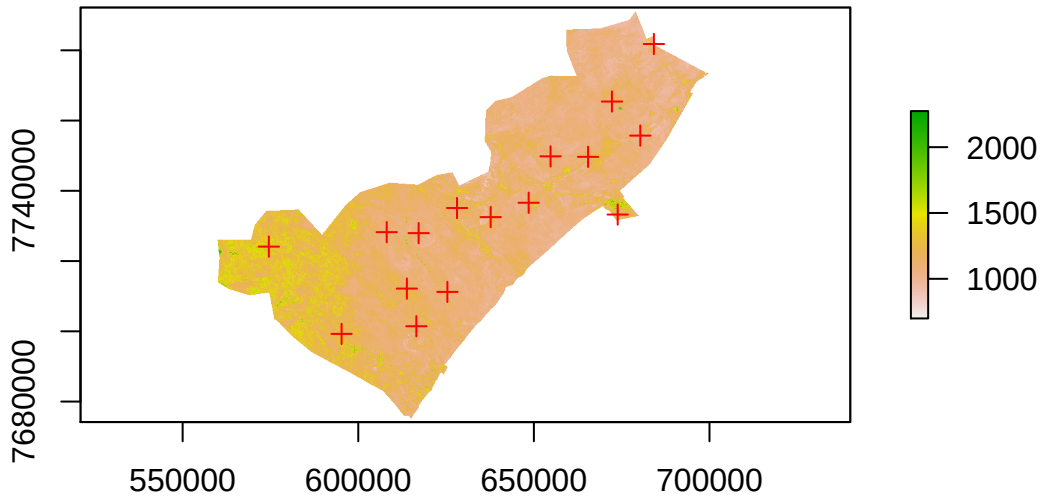
```
## class       : Extent
## xmin        : 574538.1
## xmax        : 684170.6
## ymin        : 7699266
## ymax        : 7781799
```

Set extent of the training data to match covs

```
trainData@bbox <- bbox(Otjo_BushArea)
```

Let's plot the points ontop of `layer 3` of the raster stack

```
plot(rstack[[3]])
plot(trainData, add=TRUE, col = "red", pch = 3)
```



Mask, read and stack the covariates

```
covs <- mask(rstack, Otjo_BushArea)
# writeRaster(covs, filename = "Otji_Stack_2016.tif", format = "GTiff",
# overwrite = TRUE)
```

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 training dataset, remove NAs and write the dataset as a .csv

```
names(trainData) <- c("waypoint_no","latitude","longitude","shrubs_less1.5",
                      "shrubs_more1.5_no_stem","shrubs_more1.5_stem",
                      "shrubs_more1.5", "shrubs_all", "crown_cover","NDVI",
                      "band2","band3","band4","band5","band6","band7")
trainData@data<-trainData@data[complete.cases(trainData@data),]
write.csv(trainData@data, file = "Otji_MF_trainData.csv",row.names=FALSE)
```

Randomly select index numbers and use that for splitting the data Set 75% as training nad 25% as test data

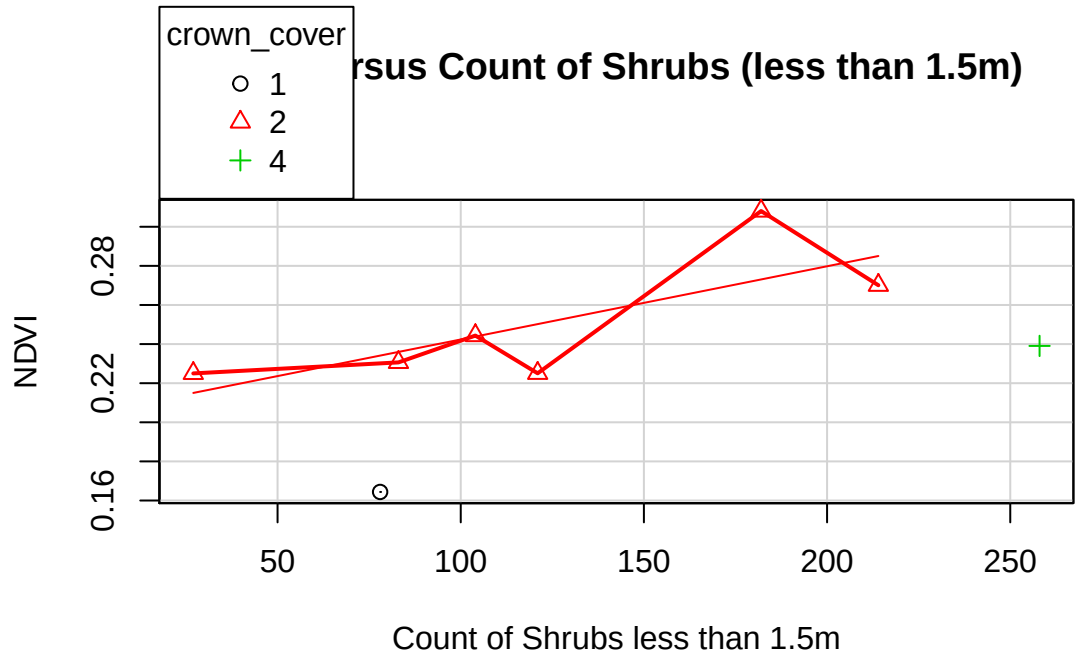
```
trainIndex=sample(1:nrow(trainData@data),size=0.75*nrow(trainData@data))
trainingSet=trainData@data[trainIndex,]
testingSet=trainData@data[-trainIndex,]
```

Let's plot some of the variables

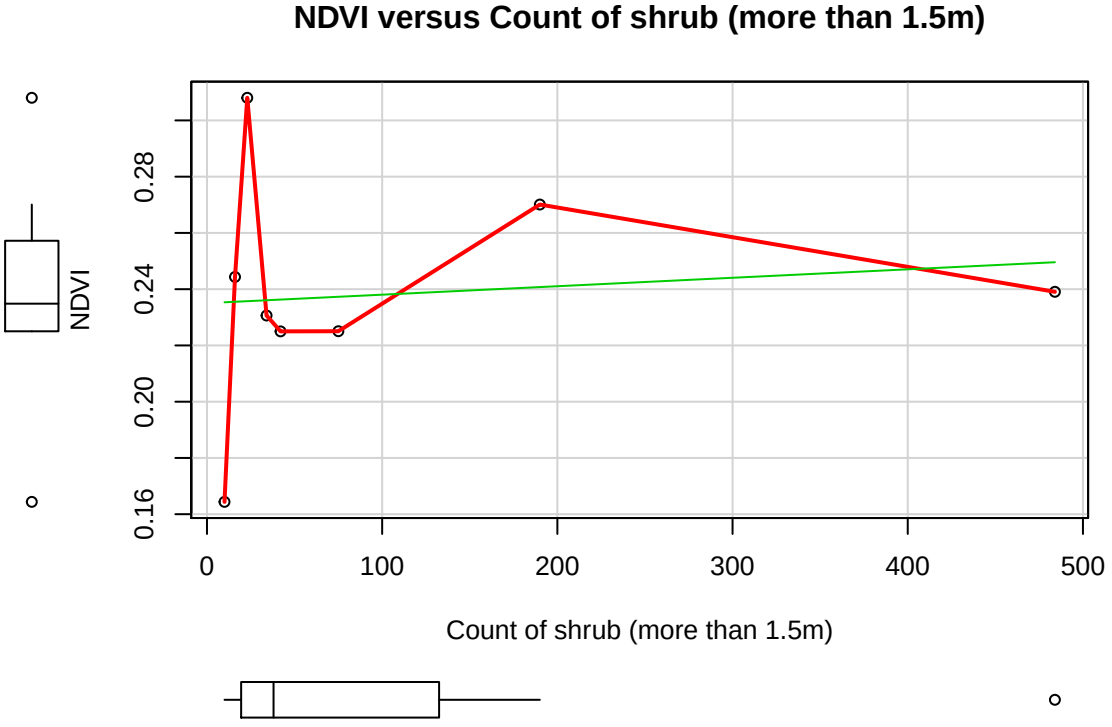
```
scatterplot(NDVI ~ shrubs_less1.5|crown_cover, data=trainingSet,
            xlab="Count of Shrubs less than 1.5m",ylab="NDVI",
            main="NDVI versus Count of Shrubs (less than 1.5m)")
```

```
## Warning in smoother(.x[subs], .y[subs], col = col[i], log.x =
## logged("x"), : could not fit smooth
```

```
## Warning in smoother(.x[subs], .y[subs], col = col[i], log.x =
## logged("x"), : could not fit smooth
```

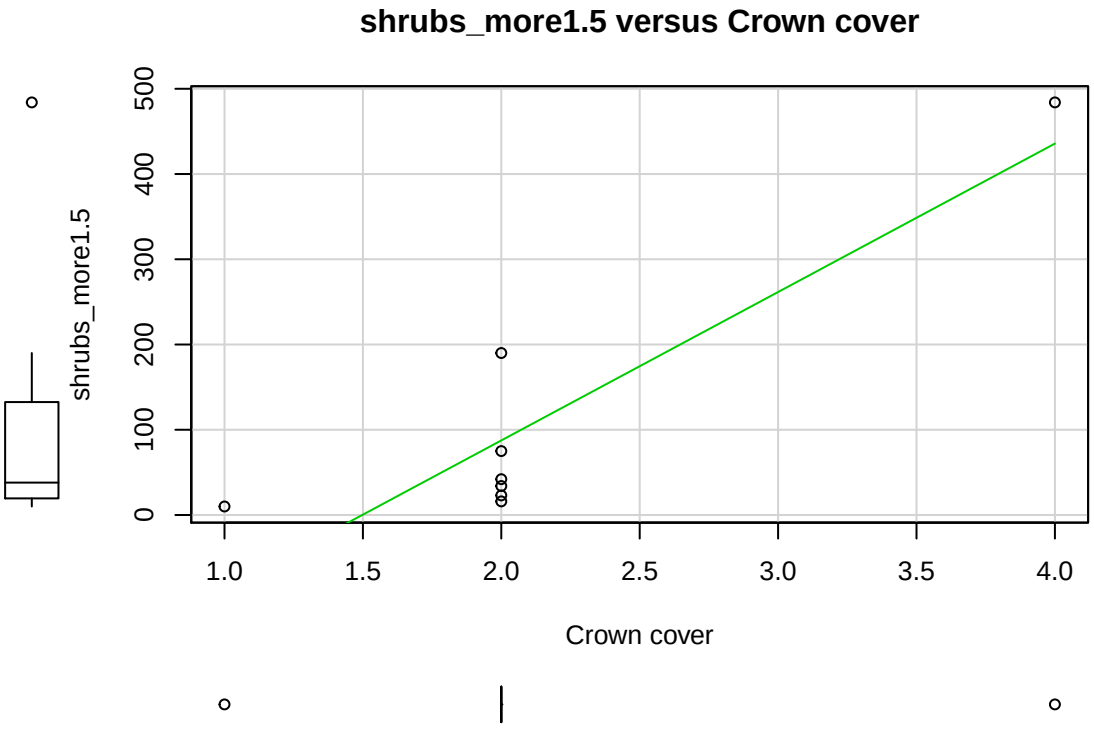



```
scatterplot(NDVI ~ shrubs_more1.5, data=trainingSet,
            xlab="Count of shrub (more than 1.5m)",ylab="NDVI",
            main="NDVI versus Count of shrub (more than 1.5m)")
```



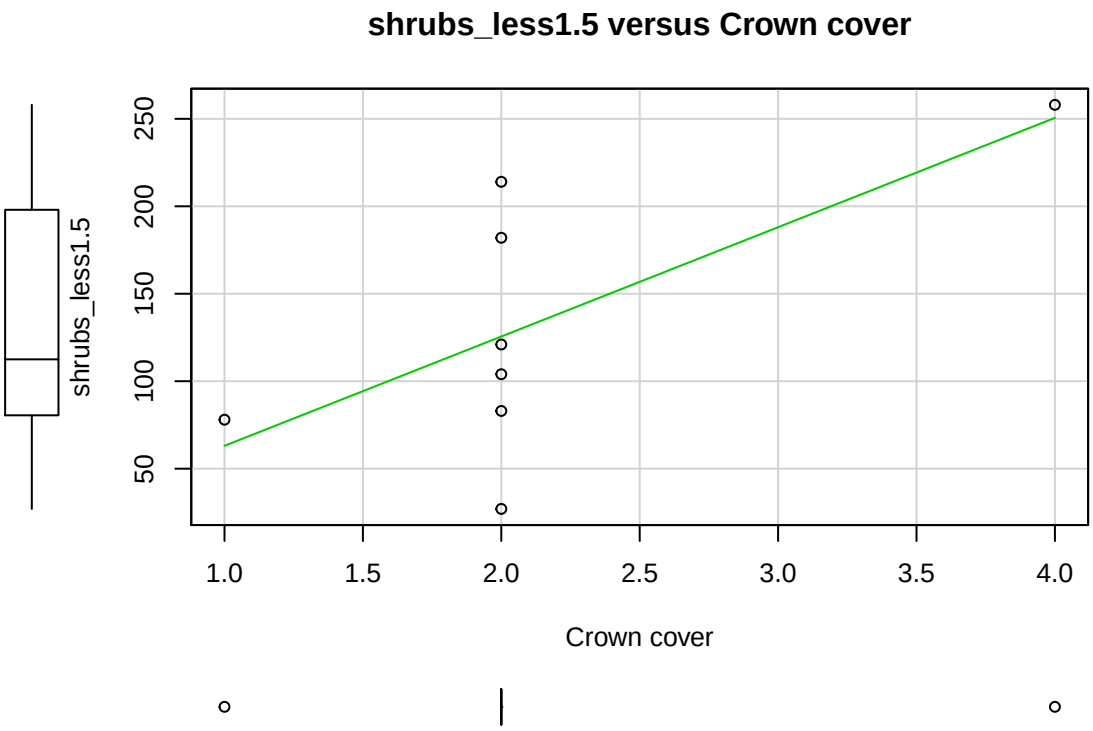
```
scatterplot(shrubs_more1.5 ~ crown_cover, data=trainingSet,
            xlab="Crown cover",ylab="shrubs_more1.5",
            main="shrubs_more1.5 versus Crown cover")
```

```
## Warning in smoother(.x, .y, col = col[2], log.x = logged("x"), log.y =
## logged("y"), : could not fit smooth
```




```
scatterplot(shrubs_less1.5 ~ crown_cover, data=trainingSet,
            xlab="Crown cover",ylab="shrubs_less1.5",
            main="shrubs_less1.5 versus Crown cover")
```

```
## Warning in smoother(.x, .y, col = col[2], log.x = logged("x"), log.y =
## logged("y"), : could not fit smooth
```



You can now set up the model using randomly sampled data

```
accuracies1<-c()
accuracies2<-c()
accuracies3<-c()
```

Construct the rf model for shrubs less than 1.5m

```
model1 <- randomForest(x=trainingSet[,c(9:10)],
                       y=trainingSet[,"shrubs_less1.5"],
                       ntree=2000, proximity=TRUE, importance=TRUE)
```

Predict density for shrubs less than 1.5m

```
otji_bd1<-predict(covs, model1, filename="otji_bd1", format='GTiff',
                  type="response", index=1, na.rm=TRUE, progress="window",
                  overwrite=TRUE)
```

```
## Loading required namespace: tcltk
```

```
plot(otji_bd1, main="Density for shrubs less than 1.5m", axes=FALSE)
```


```
prediction1 <- predict(model1, testingSet)
testingSet$rightPred1 <- prediction1 == testingSet$shrubs_less1.5
t1<-table(prediction1, testingSet$shrubs_less1.5)
print(t1)
```

```
##
## prediction1        46 48 52
##   80.817694047619   0  0  1
##   87.2979928571428  1  0  0
##   88.8365011904762  0  1  0
```

```
accuracy1 <- sum(testingSet$rightPred1)/nrow(testingSet)
accuracies1 <- c(accuracies1,accuracy1)
print(accuracy1)
```

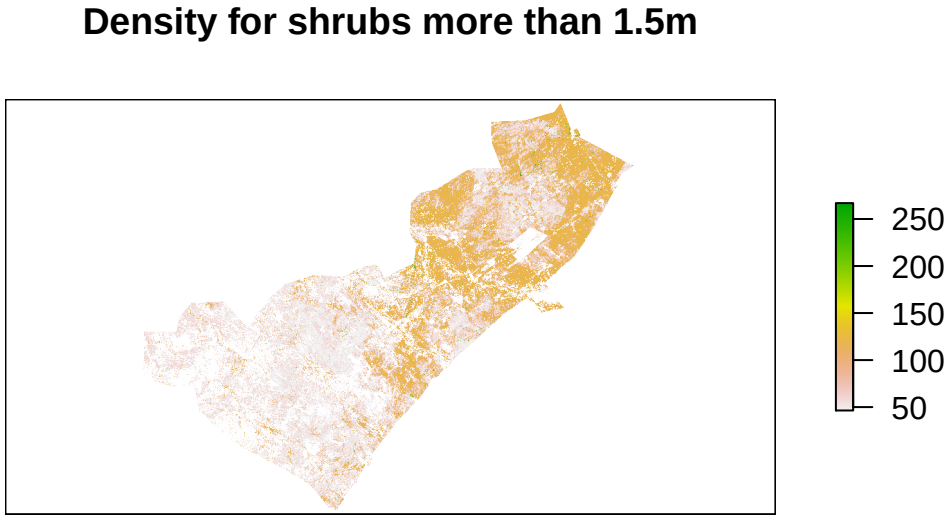
```
## [1] 0
```

Construct the rf model for shrubs more than 1.5m

```
model2 <- randomForest(x=trainingSet[,c(9:10)],
                       y=trainingSet[,"shrubs_more1.5"],
                       ntree=2000, proximity=TRUE, importance=TRUE)
```

Predict density for shrubs more than 1.5m

```
otji_bd2<-predict(covs, model2, filename="otji_bd2", format='GTiff',
                  type="response", index=1, na.rm=TRUE, progress="window",
                  overwrite=TRUE)
plot(otji_bd2, main="Density for shrubs more than 1.5m", axes=FALSE)
```




```
prediction2 <- predict(model2, testingSet)
testingSet$rightPred2 <- prediction2 == testingSet$shrubs_more1.5
t2<-table(prediction2, testingSet$shrubs_more1.5)
print(t2)
```

```
##
## prediction2         7 14 35
##   46.4059178571428 1  0  0
##   58.2383619047619 0  1  0
##   62.1545178571429 0  0  1
```

```
accuracy2 <- sum(testingSet$rightPred2)/nrow(testingSet)
accuracies2 <- c(accuracies2,accuracy2)
print(accuracy2)
```

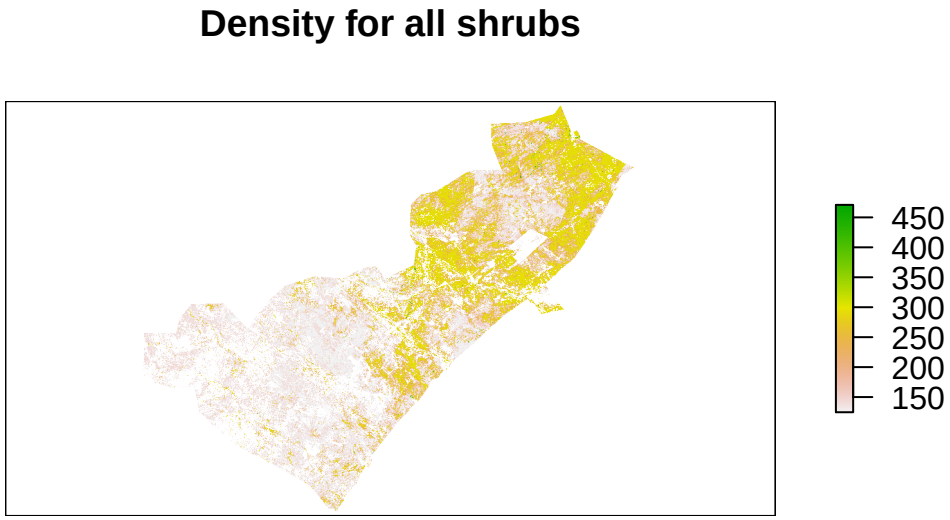
```
## [1] 0
```

Construct the rf model for for all shrubs

```
model3 <- randomForest(x=trainingSet[,c(9:10)],
                       y=trainingSet[,"shrubs_all"],
                       ntree=2000, proximity=TRUE, importance=TRUE)
```

Predict density for all shrubs

```
otji_bd3<-predict(covs, model3, filename="otji_bd3", format='GTiff',
                  type="response", index=1, na.rm=TRUE, progress="window",
                  overwrite=TRUE)
plot(otji_bd3, main="Density for all shrubs", axes=FALSE)
```




```
prediction3 <- predict(model3, testingSet)
testingSet$rightPred3 <- prediction3 == testingSet$shrubs_all
t3<-table(prediction3, testingSet$shrubs_all)
print(t3)
```

```
##
## prediction3        59 60 83
##   124.675311904762  1  0  0
##   141.808941666667  0  1  0
##   145.994144047619  0  0  1
```

```
accuracy3 <- sum(testingSet$rightPred3)/nrow(testingSet)
accuracies3 <- c(accuracies3,accuracy3)
print(accuracy3)
```

```
## [1] 0
```

print all the models

```
print(model1)
```

```
##
## Call:
##  randomForest(x = trainingSet[, c(9:10)], y = trainingSet[, "shrubs_less1.5"],      ntree =
##                Type of random forest: regression
##                      Number of trees: 2000
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 5150.066
##                     % Var explained: 2.67
```

```
print(model2)
```

```
##
## Call:
##  randomForest(x = trainingSet[, c(9:10)], y = trainingSet[, "shrubs_more1.5"],      ntree =
##                Type of random forest: regression
##                      Number of trees: 2000
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 30790.38
##                     % Var explained: -33.65
```

```
print(model3)
```

```
##
## Call:
##  randomForest(x = trainingSet[, c(9:10)], y = trainingSet[, "shrubs_all"],      ntree = 200
##                Type of random forest: regression
##                      Number of trees: 2000
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 56050.75
##                     % Var explained: -23.23
```

As a bonus you can check the amount of time you spent conducting this analysis

```
timeDiff <- Sys.time() - startTime
cat("\nProcessing time", format(timeDiff), "\n")
```

```
##
## Processing time 45.08945 mins
```