Bush density mapping

CIAT

31 October, 2016, 23:22

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.
5. Use the RF model to predict bush density.
6. Test model accuracy.

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

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

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)
```

Set the start of data processing.

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

```
## Start time 2016-10-31 23:22:04
```

Set working directory.

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

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.

```r
.packages = c("sp","rgdal","raster","randomForest","plyr","xlsx","xlsxjars",
              "dplyr","caret","car", "e1071","snow")
.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

## Loading required package: e1071

##
## Attaching package: 'e1071'

## The following object is masked from 'package:raster':
##
##     interpolate

## Loading required package: snow
```

```
## [[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
##
## [[11]]
## [1] TRUE
##
## [[12]]
## [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
```

Reading your data

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 and rename trainData fields

```
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"
```

```
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

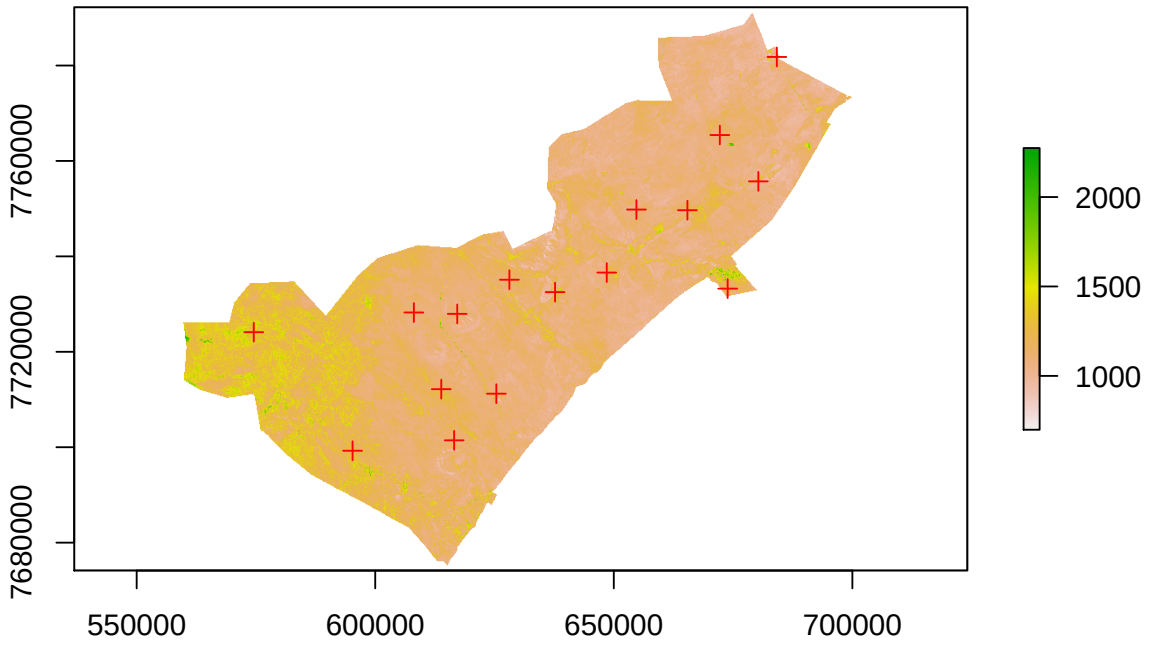
```
Otjo_BushArea <- raster("Raster_data/Other_data/Otji_BushArea_2016.tif")
```

Set extent of the training data to match covs

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

Plot the points on top 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)
```

Assign raster values to the training data.

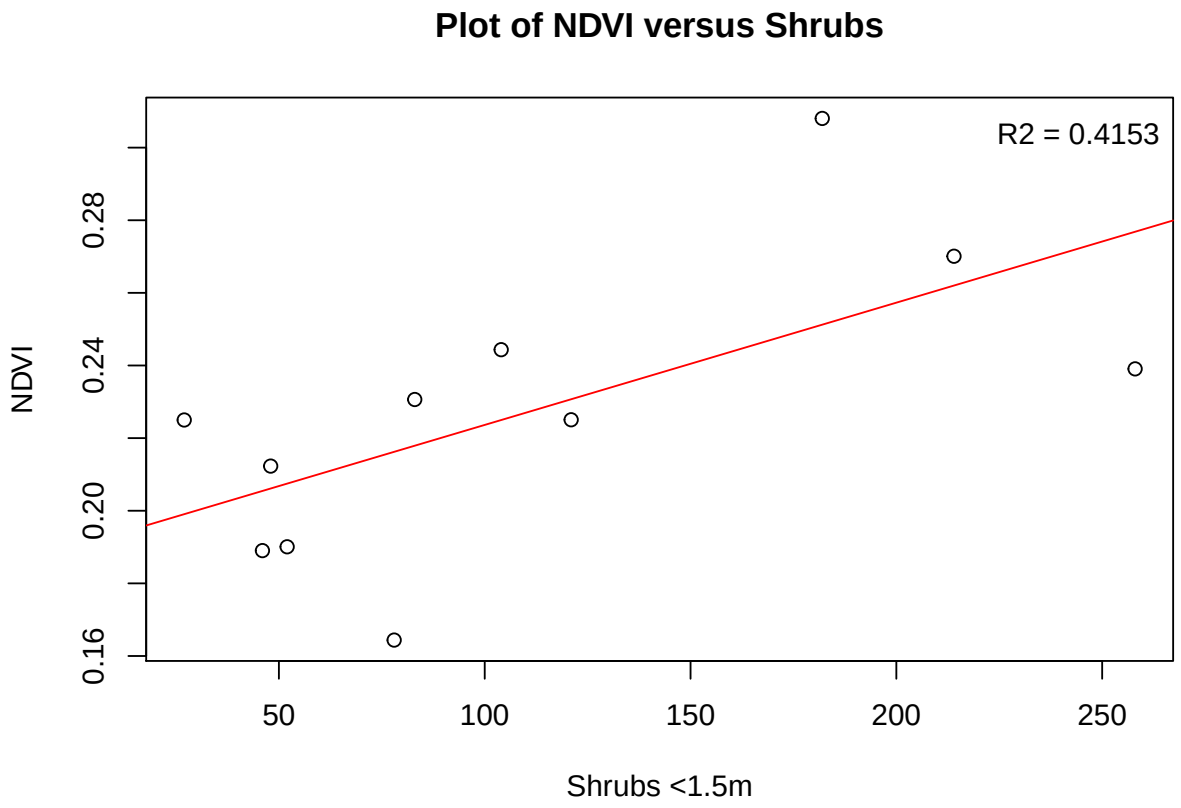
```
v<-as.data.frame(extract(covs,trainData))
trainData@data=data.frame(trainData@data, v[match(rownames(trainData@data),
                                                  rownames(v)),])
```

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

```
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)
```

Let's generate a plot of NDVI versus shrubs and add the R squared value on the plot.

```
with(trainData@data, plot(shrubs_less1.5,NDVI,
                          xlab="Shrubs <1.5m", ylab="NDVI",
                          main="Plot of NDVI versus Shrubs"))
abline(fit <- lm(NDVI~shrubs_less1.5, data = trainData@data), col='red')
legend("topright", bty="n", legend=paste("R2 =",
                                         format(summary(fit)$r.squared,
                                                digits=4)))
```



Fitting the Random Forest regression models

You can now fit the models using the 'train' function from the 'caret' package. i.e. Specify the model as a formula with the dependent variable (i.e., count of shrubs).

Use Bootstrap resampling method to estimate model accuracy. This method involves taking random samples from the dataset (with re-selection) against which to evaluate the model. In aggregate, the results provide an indication of the variance of the models performance.

```
tcontrol1 <- trainControl(method="boot", number=100)
model1 <- train(shrubs_less1.5~crown_cover+NDVI,method='rf',
                trControl = tcontrol1, data=trainData@data)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): The response
## has five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): The response
## has five or fewer unique values. Are you sure you want to do regression?

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
tcontrol3 <- trainControl(method="boot", number=100)
model3 <- train(shrubs_all~crown_cover+NDVI,method='rf',
                trControl = tcontrol3, data=trainData@data)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): The response
## has five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): The response
## has five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): The response
## has five or fewer unique values. Are you sure you want to do regression?

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

Next, before you predict the models, you can print out the RMSE and Rsquared . R squared is a number that indicates the proportion of the variance in the dependent variable that is predictable from the independent variable.

```
print(model1)
```

```
## Random Forest
##
## 11 samples
## 15 predictors
##
## No pre-processing
## Resampling: Bootstrapped (100 reps)
## Summary of sample sizes: 11, 11, 11, 11, 11, 11, ...
## Resampling results:
##
##   RMSE      Rsquared
##   60.55145  0.6287316
##
## Tuning parameter 'mtry' was held constant at a value of 2
##
```

```
print(model3)
```

```
## Random Forest
##
## 11 samples
## 15 predictors
##
## No pre-processing
## Resampling: Bootstrapped (100 reps)
## Summary of sample sizes: 11, 11, 11, 11, 11, 11, ...
## Resampling results:
##
##   RMSE      Rsquared
##   186.4798  0.5044575
##
## Tuning parameter 'mtry' was held constant at a value of 2
##
```

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).

```
beginCluster()
```

```
## 4cores detected
```

```
prediction1 <- clusterR(covs, raster::predict, args = list(model = model1))
prediction3 <- clusterR(covs, raster::predict, args = list(model = model3))
endCluster()
```

Compute the density for shrubs above 1.5m. We can do this by substracting shrubs less than 1.5m from all shrubs.

```
prediction2 <- prediction3 - prediction1
```

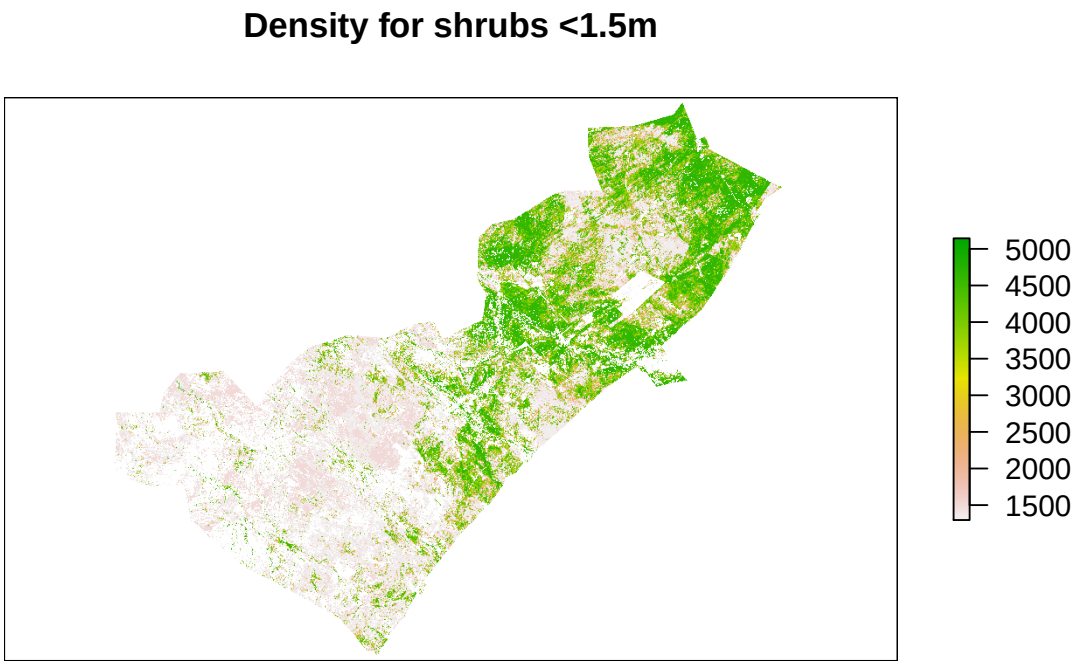
Multiply the output rasters by 25 to convert the units from shrubs/0.04ha to shrubs/1ha, round raster values to whole numbers and save the predicted images as GeoTIFFs.

```
prediction1<-round(prediction1*25, digits = 0)
prediction2<-round(prediction2*25, digits = 0)
prediction3<-round(prediction3*25, digits = 0)
writeRaster(prediction1, "otji_bd1.tif", overwrite=TRUE)
writeRaster(prediction2, "otji_bd2.tif", overwrite=TRUE)
writeRaster(prediction3, "otji_bd3.tif", overwrite=TRUE)
```

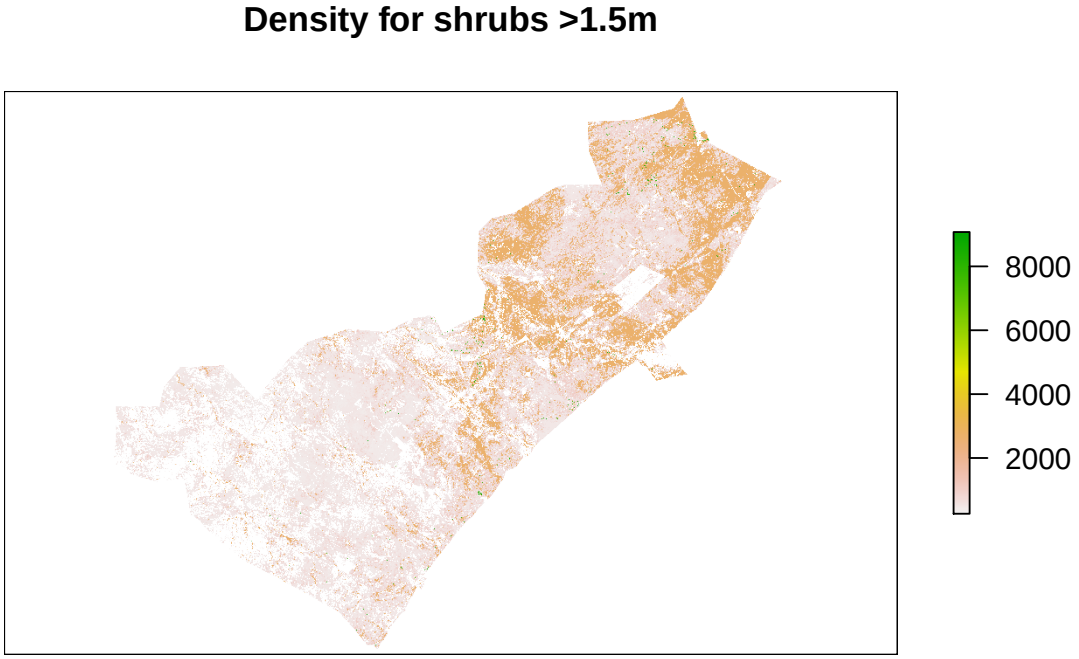
Results

Plot the three maps.

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




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




```
plot(prediction3, main="Density for shrubs", axes=FALSE)
```

Finally, check the amount of time you spent conducting this analysis

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

```
##
## Processing time 18.20064 mins
```