Bush density mapping

CIAT

06 November, 2016, 19:49

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.

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.

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

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

```
#set the random seed
set.seed(211134)
```

Set the start of data processing.

```
#set the start of data processing
startTime <- Sys.time()
cat("Start time", format(startTime),"\n")
```

```
## Start time 2016-11-06 19:49:03
```

Set working directory.

```
#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.

```
#load packages
.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: 1.1-10, (SVN revision 622)
##  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/3.3/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: C:/Users/jymutua/Documents/R/win-library/3.3/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
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
## 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:

```
#this is how you get more help on functions
help(calc)
?calc
```

Reading your data

Read in data from excel sheet

```
#read in data from excel sheet
raw.d <- 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

```
#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 columns you need

```
#create new dataframe with columns you need
raw.d<-raw.d[,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

```
#add two new columns of shrubs > 1.5 and all shrubs in general
raw.d$shrubs_more_1.5 <- apply(raw.d[,5:6], 1, sum, na.rm=TRUE)
raw.d$shrubs_all <- apply(raw.d[,4:6], 1, sum, na.rm=TRUE)
```

Round columns and create new 'data.frame with the columns you need.

```
#round columns
raw.d<-raw.d %>%
```

```
  mutate_each(funs(round(.,0)), shrubs_less_1.5, shrubs_more_1.5,
              shrubs_all)
raw.d<-raw.d[,c("Waypoint_No","Latitude","Longitude","shrubs_less_1.5",
        "shrubs_more_1.5", "shrubs_all")]
```

Compute shrubs per hectare.

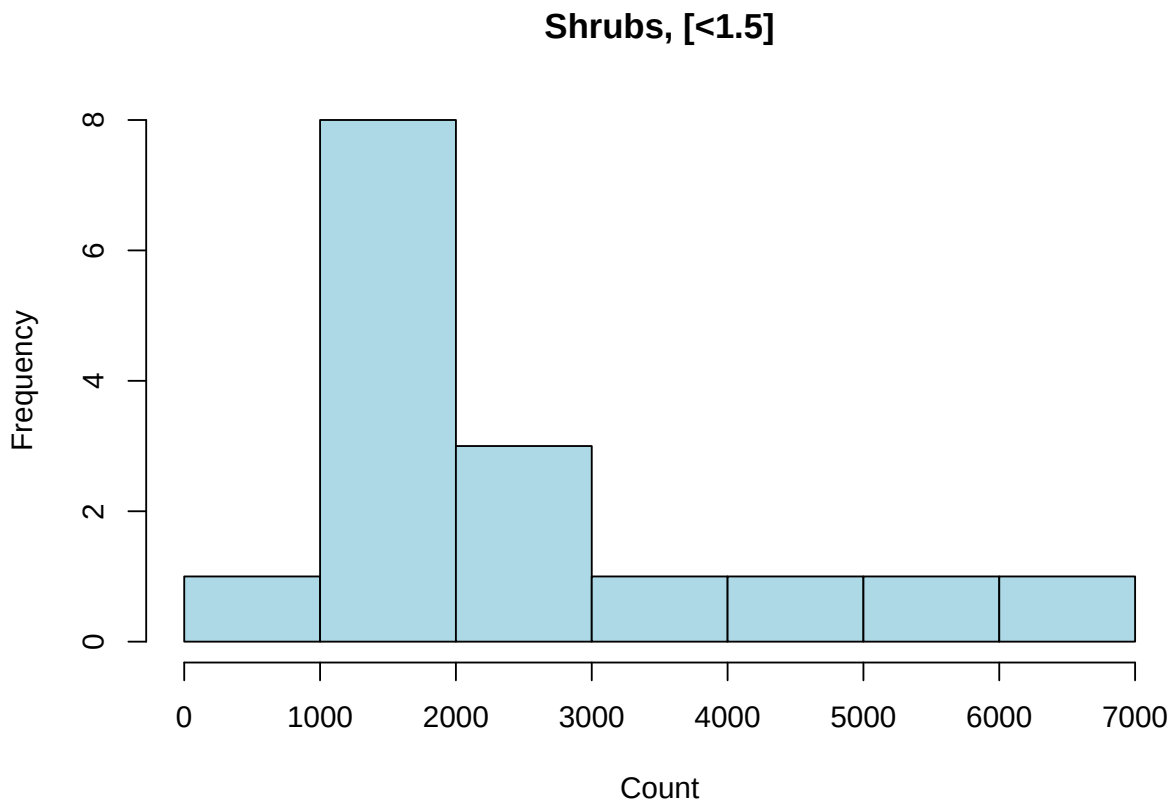
```
#compute shrubs per hectare
raw.d$shrubs_less_1.5 <- raw.d$shrubs_less_1.5*25
raw.d$shrubs_more_1.5 <- raw.d$shrubs_more_1.5*25
raw.d$shrubs_all <- raw.d$shrubs_all*25
```

Remove all NAs.

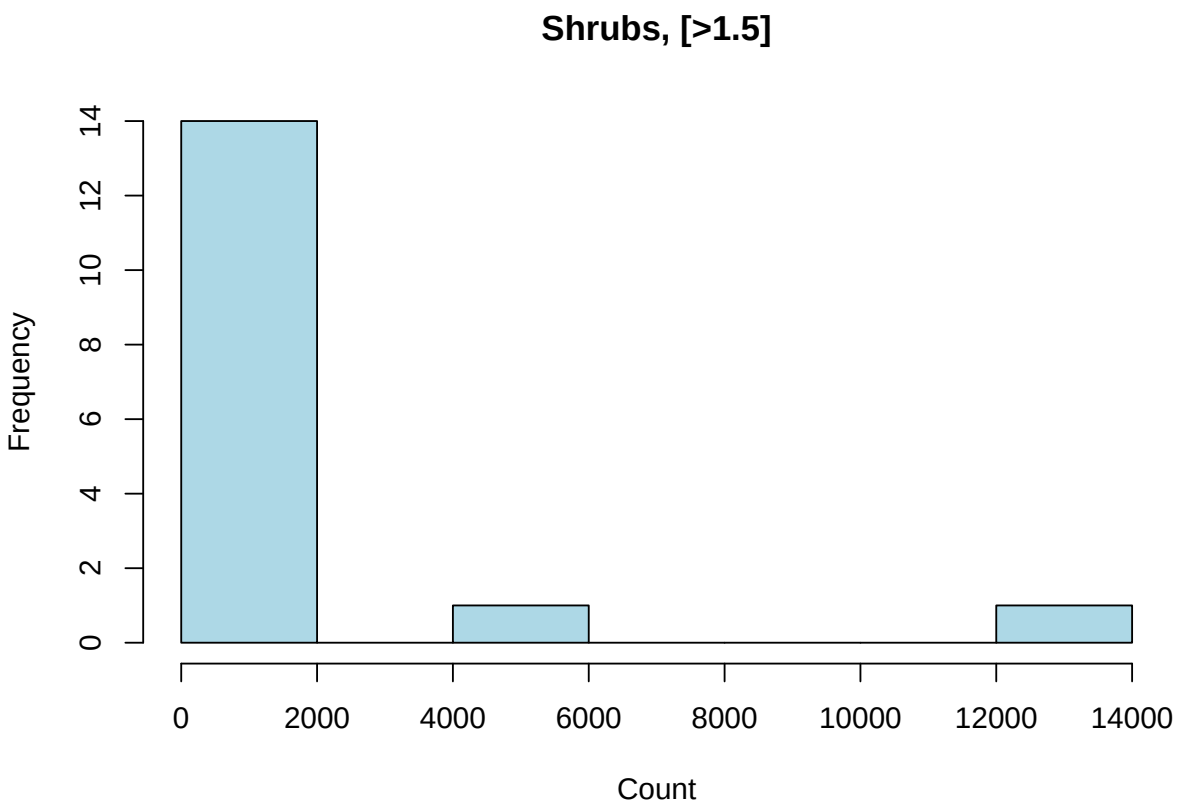
```
#remove all NAs
raw.d<-raw.d[complete.cases(raw.d),]
```

Plot histograms of the three variables

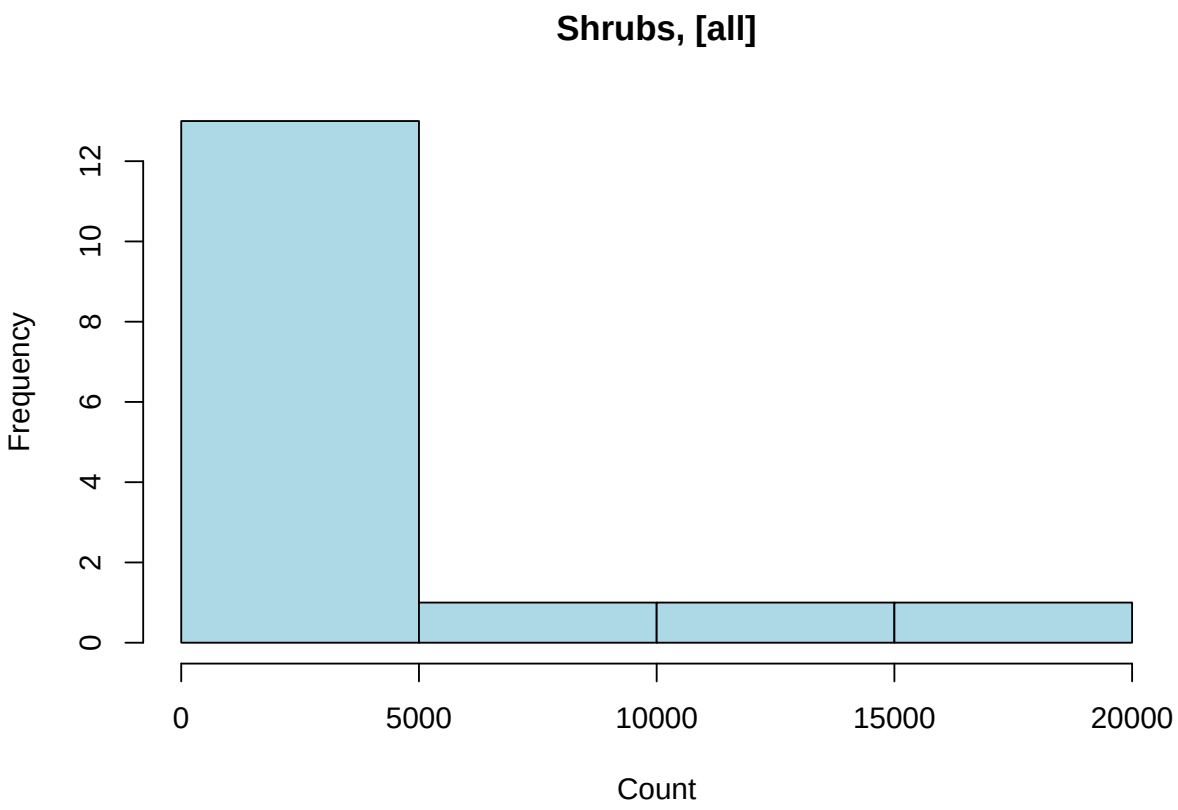
```
#plot histograms of the three variables
hist(raw.d$shrubs_less_1.5, col = "lightblue", xlab="Count", main="Shrubs, [<1.5]")
```




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




```
hist(raw.d$shrubs_all, col = "lightblue", xlab="Count", main="Shrubs, [all]")
```



Export data to .csv

```
#export data to .csv
write.csv(raw.d, 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.

```
#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 fields

```
#rename fields
names(trainData)
```

```
## [1] "Waypoint_No"     "Latitude"        "Longitude"       "shrubs_less_1.5"
## [5] "shrubs_more_1.5" "shrubs_all"
```

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

Import the rest of input data, stack and rename contents

```
#import the rest of input data, stack and rename contents
r.list<-list.files(".", pattern = ".tif$", full.names = TRUE)
r.stack <- stack(r.list)
names(r.stack) <- c("cc","ndvi","b2","b3","b4","b5","b6","b7")
```

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

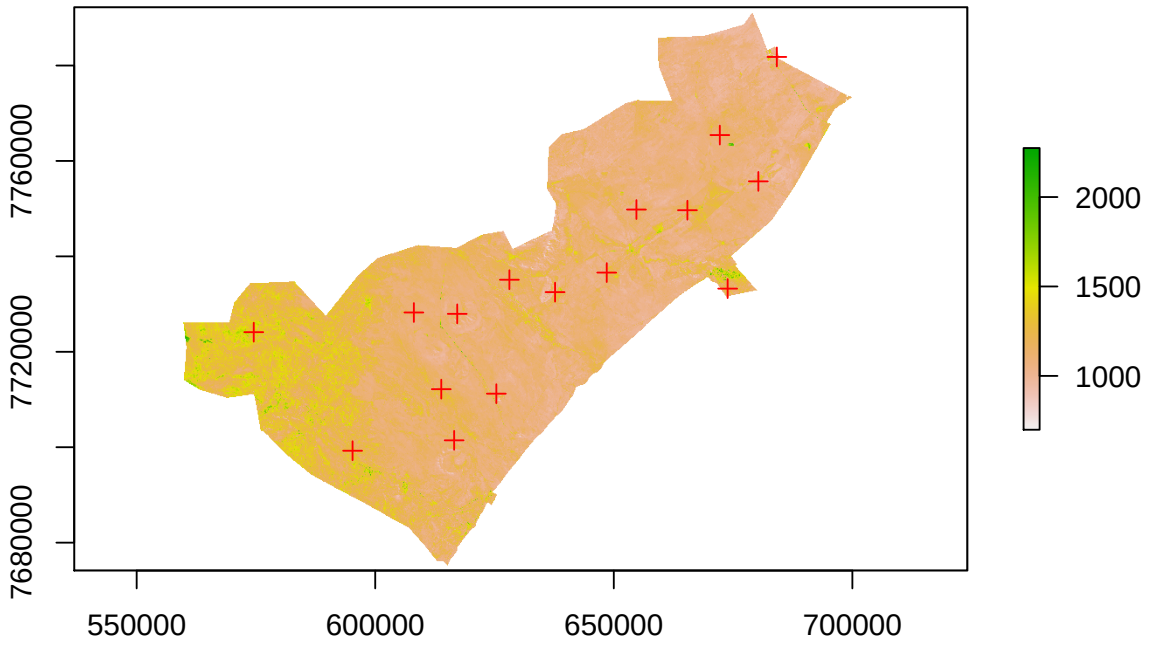
```
#import the bush area mask
o.mask <- raster("Other_data/Otji_BushArea_2016.tif")
```

Set extent of the training data to match covs

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

Plot the points on top of `layer 3` of the raster stack

```
#plot the points on top of `layer 3` of the raster stack
plot(r.stack[[3]])
plot(trainData, add=TRUE, col = "red", pch = 3)
```



Mask, read and stack the covariates. Remove NA values (otherwise RF cannot predict)

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

Assign raster values to the training data.

```
#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

```
#rename fields in the training dataset, remove NAs, write the dataset
names(trainData) <- c("waypoint.no","lat","lon","shrubs.l1.5","shrubs.g1.5",
                      "shrubs.all","cc","ndvi","b2","b3","b4","b5","b6","b7")
trainData@data<-trainData@data[complete.cases(trainData@data),]
write.csv(trainData@data, file = "Otji_MF_trainData.csv",row.names=FALSE)
```

Compute summary statistics.

```
#compute summary statistics
summary(trainData$shrubs.all)
```

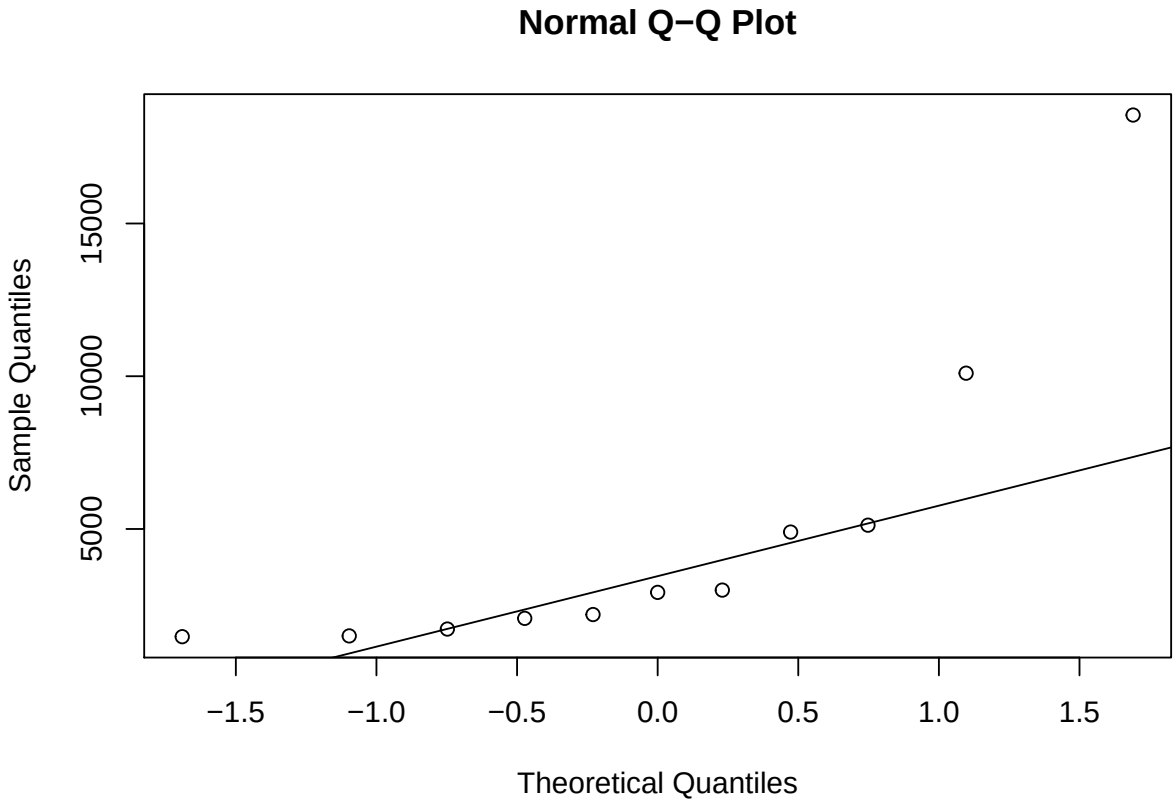
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1475    1900    2925    4870    5012   18550
```

```
skewness(trainData$shrubs.all, na.rm=T)
```

```
## [1] 1.649636
```

QQ plot.

```
#QQ plot
qqnorm(trainData$shrubs.all)
qqline(trainData$shrubs.all)
```



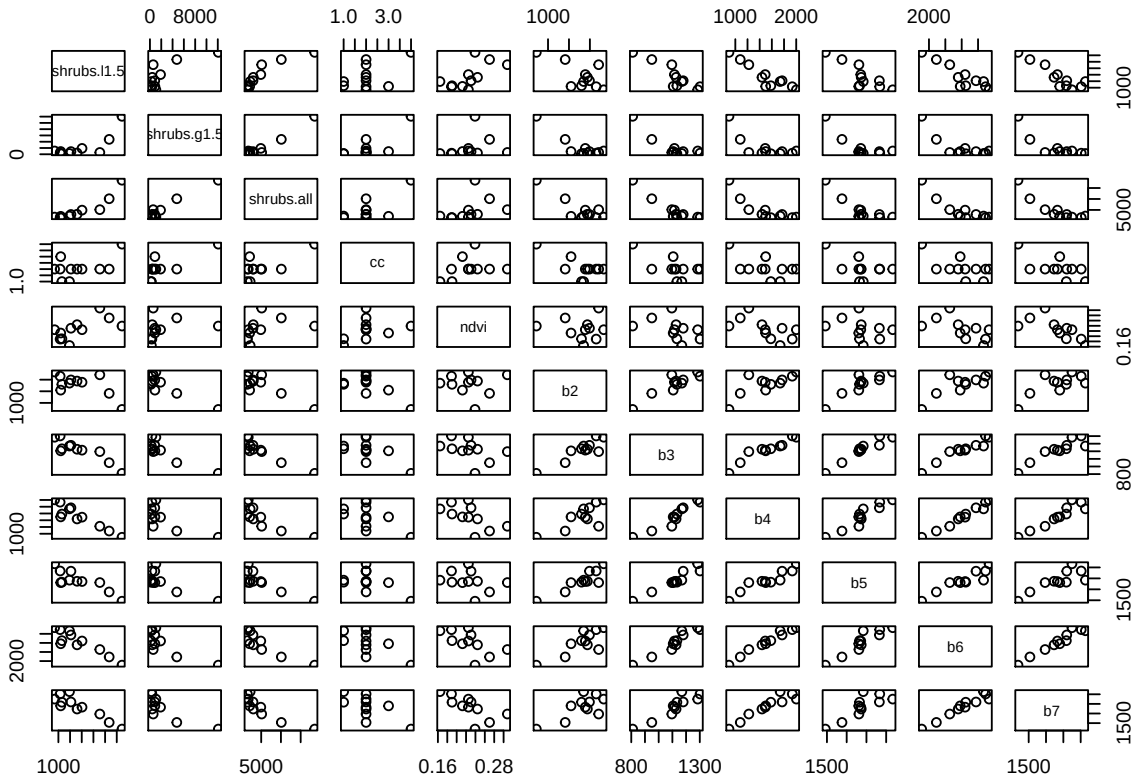
Compute correlation coefficients and plot correlations.

```
#compute correlation coefficients and plot correlations
cor(trainData@data[,4:14])
```

```
##              shrubs.l1.5 shrubs.g1.5 shrubs.all         cc        ndvi
## shrubs.l1.5    1.0000000   0.7870470  0.9062045  0.5113658  0.64446772
## shrubs.g1.5    0.7870470   1.0000000  0.9740724  0.7618735  0.24796438
## shrubs.all     0.9062045   0.9740724  1.0000000  0.7097503  0.40631237
## cc             0.5113658   0.7618735  0.7097503  1.0000000  0.32918259
## ndvi           0.6444677   0.2479644  0.4063124  0.3291826  1.00000000
## b2            -0.6673499  -0.8699912 -0.8410627 -0.6596828 -0.04182892
## b3            -0.8883771  -0.8492370 -0.9078956 -0.6004480 -0.42154722
```

```
## b4           -0.9182825  -0.7182936 -0.8291098 -0.5363318 -0.60545915
## b5           -0.8334936  -0.7572776 -0.8247357 -0.4960407 -0.29955664
## b6           -0.8988966  -0.7742281 -0.8603397 -0.6219464 -0.59969871
## b7           -0.9030470  -0.7987772 -0.8786887 -0.6675928 -0.70287392
##                      b2         b3         b4         b5         b6
## shrubs.l1.5 -0.66734993 -0.8883771 -0.9182825 -0.8334936 -0.8988966
## shrubs.g1.5 -0.86999115 -0.8492370 -0.7182936 -0.7572776 -0.7742281
## shrubs.all  -0.84106273 -0.9078956 -0.8291098 -0.8247357 -0.8603397
## cc          -0.65968278 -0.6004480 -0.5363318 -0.4960407 -0.6219464
## ndvi        -0.04182892 -0.4215472 -0.6054592 -0.2995566 -0.5996987
## b2           1.00000000  0.8998662  0.7552128  0.8846406  0.7948454
## b3           0.89986622  1.0000000  0.9515501  0.9578004  0.9568856
## b4           0.75521283  0.9515501  1.0000000  0.9400076  0.9851810
## b5           0.88464059  0.9578004  0.9400076  1.0000000  0.9237968
## b6           0.79484537  0.9568856  0.9851810  0.9237968  1.0000000
## b7           0.72027622  0.9121251  0.9365657  0.8204991  0.9682497
##                      b7
## shrubs.l1.5 -0.9030470
## shrubs.g1.5 -0.7987772
## shrubs.all  -0.8786887
## cc          -0.6675928
## ndvi        -0.7028739
## b2           0.7202762
## b3           0.9121251
## b4           0.9365657
## b5           0.8204991
## b6           0.9682497
## b7           1.0000000
```

```
pairs(trainData@data[,4:14])
```



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

```
#correlate count of shrubs with NDVI and Landsat 8 band 2-7
cor(trainData@data$shrubs.l1.5,trainData@data$ndvi)
```

```
## [1] 0.6444677
```

```
cor(trainData@data$shrubs.all,trainData@data$ndvi)
```

```
## [1] 0.4063124
```

```
cor(trainData@data$shrubs.l1.5,trainData@data$cc)
```

```
## [1] 0.5113658
```

```
cor(trainData@data$shrubs.all,trainData@data$cc)
```

```
## [1] 0.7097503
```

```
cor(trainData@data$shrubs.l1.5,trainData@data$b2)
```

```
## [1] -0.6673499
```

```
cor(trainData@data$shrubs.all,trainData@data$b2)
```

```
## [1] -0.8410627
```

```
cor(trainData@data$shrubs.l1.5,trainData@data$b3)
```

```
## [1] -0.8883771
```

```
cor(trainData@data$shrubs.all,trainData@data$b3)
```

```
## [1] -0.9078956
```

```
cor(trainData@data$shrubs.l1.5,trainData@data$b6)
```

```
## [1] -0.8988966
```

```
cor(trainData@data$shrubs.all,trainData@data$b6)
```

```
## [1] -0.8603397
```

```
cor(trainData@data$shrubs.l1.5,trainData@data$b7)
```

```
## [1] -0.903047
```

```
cor(trainData@data$shrubs.all,trainData@data$b7)
```

```
## [1] -0.8786887
```

Select covariates based on correlation analysis and save as a 'data.frame'.

```
#select covariates based on correlation analysis
d <- trainData@data[,c("waypoint.no","lat","lon","shrubs.l1.5","shrubs.all",
                       "cc", "ndvi")]
names(d)
```

```
## [1] "waypoint.no" "lat"         "lon"         "shrubs.l1.5" "shrubs.all"
## [6] "cc"          "ndvi"
```

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.

```
#fit the model for shrubs <1.5m
tcontrol1 <- trainControl(method="boot", number=100)
model1 <- train(shrubs.l1.5~cc+ndvi,method='rf', trControl = tcontrol1, data=d)
```

```
## 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 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.
```

```r
#fit the model for all shrubs
tcontrol3 <- trainControl(method="boot", number=100)
model3 <- train(shrubs.all~cc+ndvi,method='rf',trControl = tcontrol3, data=d)
```

```
## 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 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 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.

```r
#print models
print(model1)
```

```
## Random Forest
##
## 11 samples
##  2 predictor
```

```
##
## No pre-processing
## Resampling: Bootstrapped (100 reps)
## Summary of sample sizes: 11, 11, 11, 11, 11, 11, ...
## Resampling results:
##
##   RMSE      Rsquared
##   1583.569  0.599454
##
## Tuning parameter 'mtry' was held constant at a value of 2
##
```

```
print(model3)
```

```
## Random Forest
##
## 11 samples
##  2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (100 reps)
## Summary of sample sizes: 11, 11, 11, 11, 11, 11, ...
## Resampling results:
##
##   RMSE      Rsquared
##   4219.018  0.4452927
##
## 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).

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

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

```
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.

```
#compute the density for shrubs >1.5m
prediction2 <- prediction3 - prediction1
```

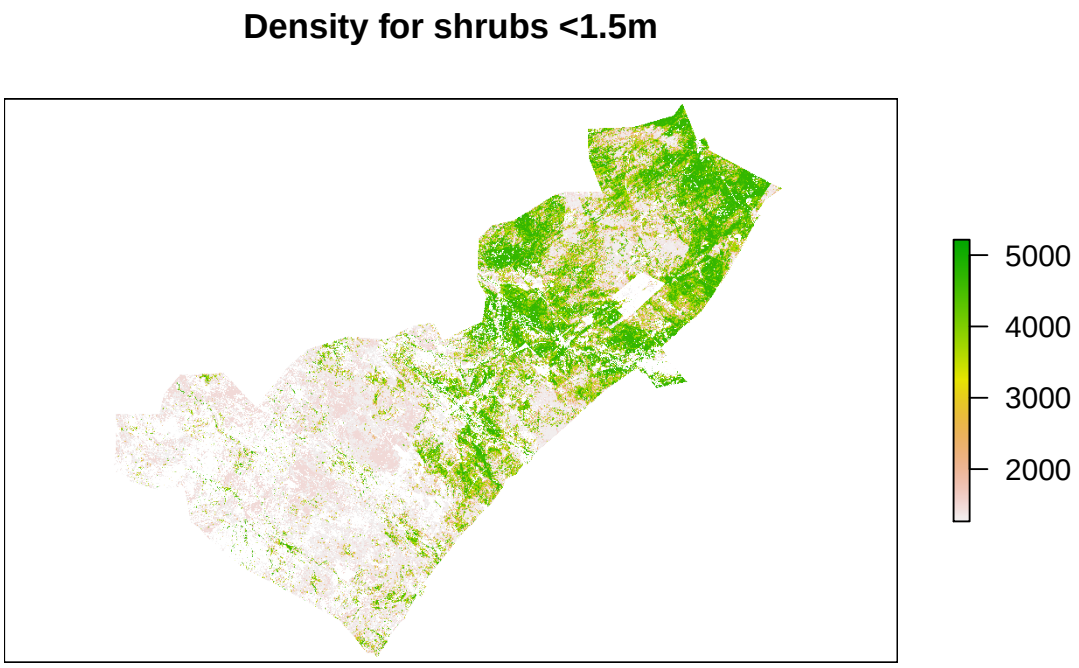
Round raster values to whole numbers and save the predicted images as GeoTIFFs.

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

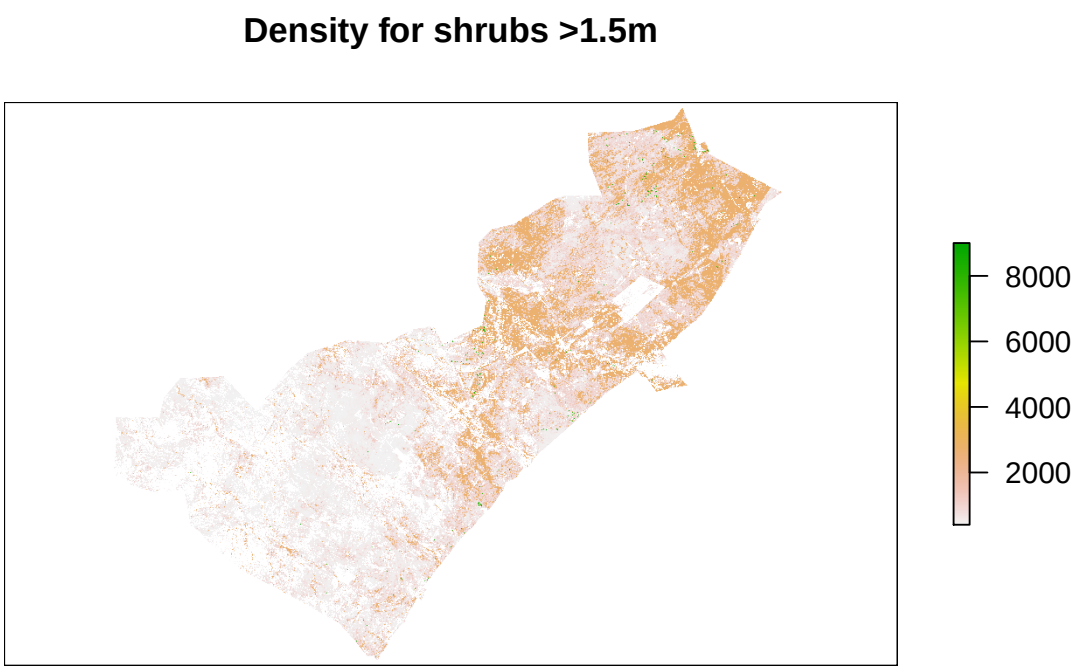
Results

Plot the three maps.

```
#plot 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

```
#check amount of time spent
timeDiff <- Sys.time() - startTime
cat("\nProcessing time", format(timeDiff), "\n")
```

```
##
## Processing time 6.431001 mins
```