Data Carpentry Semester Project

Christine Swanson

December 7, 2016

```r
knitr::opts_chunk$set(message = FALSE)
knitr::opts_chunk$set(warning = FALSE)
```

The goal of this project is to produce a random forest classification for a scene in Tocantins, Brazil. This is an introductory project for my dissertation research which will focus on impacts of dams on floodplains and riparian forests. This part of the project helped me learn more about random forest classification as well as how to work with spatial data in R. I used Landsat 8 data which had been atmospherically corrected. These data were collected on August 5, 2013. The full project can be found on [Github] (https://github.com/thatsciencegal/RS_semester_proj2).

All code was constructed in R. Sample points for the random forest classification were selected first in Google Earth then refined in a 5, 4, 3 composite in ArcMap 10.4. These points were saved as a shape file and bought into R. The random forest classifier was built with the randomForest package. However, the predictions portion of this package did not produce the map I needed in order to look at the predicted values in a raster. I used the caret package's prediction utility to create a raster of predicted values for the region.

The first step in this project is to load the libraries.

```r
library(rgdal)
library(raster)
library(RStoolbox)
library(caret)
library(randomForest)
library(e1071)
library(snow)
```

Next, I created a list of files for the Landsat 8 raster data.

```r
rasters_2013217 <- list.files(path = "./Data",
                              pattern = "LC82220672013217LGN00_B.*.TIF",
                              full.names = TRUE)
```

I used the file list to create a stack of rasters for the different image bands.

```r
l8_2013217 <- stack(rasters_2013217)
```

In order to perform a tasseled cap analysis (which I wanted to include in my random forest), I subset the rasters which I would include in the TCA.

```r
l8_2013217_tca_stack <- stack(rasters_2013217[c(2, 3, 4, 5, 6, 7)])
```

I also isolated the rasters I would use to calculate NDVI.

```r
l8_2013217_nir <- stack(rasters_2013217[c(5)])
l8_2013217_red <- stack(rasters_2013217[c(4)])
```

I then renamed the raster layers in the original raster stack and the TCA stack to make the random forest analysis code easier to read.

```r
names(l8_2013217) <- c(paste0("B", 1:7, coll = ""), "B9")
names(l8_2013217_tca_stack) <- c(paste0("B", 2:7, coll = ""))
```

Next, I calculated the NDVI and renamed the NDVI layer.

```r
l8_2013217_ndvi <- overlay(l8_2013217_red, l8_2013217_nir, fun = function(Red, NIR){
  (NIR-Red) / (NIR+Red)
})

names(l8_2013217_ndvi) <- c("NDVI")
```

I then ran the TCA and renamed the resulting layers

```r
l8_2013217_tca <- tasseledCap(l8_2013217_tca_stack, "Landsat8OLI")
names(l8_2013217_tca) <- c("Brightness", "Greenness", "Wetness")
```

I loaded the elevation data for the area. Elevation was split into two different rasters, so I joined them with the mosaic utility in ArcGIS. The resulting raster, clipped to my study area, is loaded here then reprojected to match the projection in the original Landsat data.

```r
elev <- raster("./Data/elev_mos_clp1.tif")
new_projection <- "+proj=utm +zone=22 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0"
elev_utm22 <- projectRaster(elev, crs=new_projection)
```

The elevation raster also needed to be resampled to match the pixel size of the Landsat data.

```r
elev_utm22_resamp <- resample(elev_utm22, l8_2013217, method = "bilinear")
```

The elevation raster was also renamed for easier use within the random forest algorithm.

```r
names(elev_utm22_resamp) <- c("elev")
```

I used the elevation raster to calculate the slope for the region.

```r
l8_2013217_slope <- terrain(elev_utm22_resamp, opt=c('slope'), unit = 'degrees')
```

I then made a new raster stack which would include all of the data for the random forest algorithm.

```r
l8_2013217_rf_data <- stack(l8_2013217, l8_2013217_ndvi,
                             elev_utm22_resamp, l8_2013217_slope, l8_2013217_tca)
```

I loaded in all of the training data that I had created in ArcGIS. The training data were spatially transformed to match the Landsat 8 data. I also created the "responsecol" variable to be used in the function which would extract the values of the different rasters for each of the different training points.

```
trainData <- readOGR(dsn = "./Data/Training_Points", layer = "TO_training_points",
                     pointDropZ = TRUE)
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "./Data/Training_Points", layer: "TO_training_points"
## with 1018 features
## It has 10 fields
```

```
trainData_utm22 <- spTransform(trainData, crs(l8_2013217))
responseCol <- "Name"
```

I used a function to extract the values of each raster in the stack at each of the different training points. These values are used in the random forest algorithm.

```
extract_pixels <- function(img, training_data){
  ##Extract training pixel values
  dfAll = data.frame(matrix(vector(), nrow = 0, ncol = length(names(img)) + 1))
  for (i in 1:length(unique(training_data[[responseCol]]))){
    category <- unique(training_data[[responseCol]])[i]
    categorymap <- training_data[training_data[[responseCol]] == category,]
    dataSet <- extract(img, categorymap)

  if(is(training_data, "SpatialPointsDataFrame")){
    dataSet <- cbind(dataSet, class = as.numeric(category))
    dfAll <- rbind(dfAll, dataSet)
    }
  if(is(training_data, "SpatialPolygonsDataFrame")){
    dataSet <- lapply(dataSet, function(x){cbind(x, class = as.numeric(rep(category, nrow(x))))})
    df <- do.call("rbind", dataSet)
    dfAll <- rbind(dfAll, df)
    }
  }
  return(dfAll)
}
```

I ran the extract pixel function using my training data and previously-created raster stack.

```
l8_2013217_pixels <- extract_pixels(l8_2013217_rf_data, trainData_utm22)
```

These training pixel data were to inform the random forest model.

```
l8_2013217_rf <- randomForest(as.factor(class) ~ B1 + B2 + B3 + B4 +
                              B5 + B6 + B7 + elev + slope + NDVI +
                              Brightness + Wetness + Greenness,
                              data = l8_2013217_pixels, mtry = 2,
                              importance = TRUE, confusion = TRUE)
```

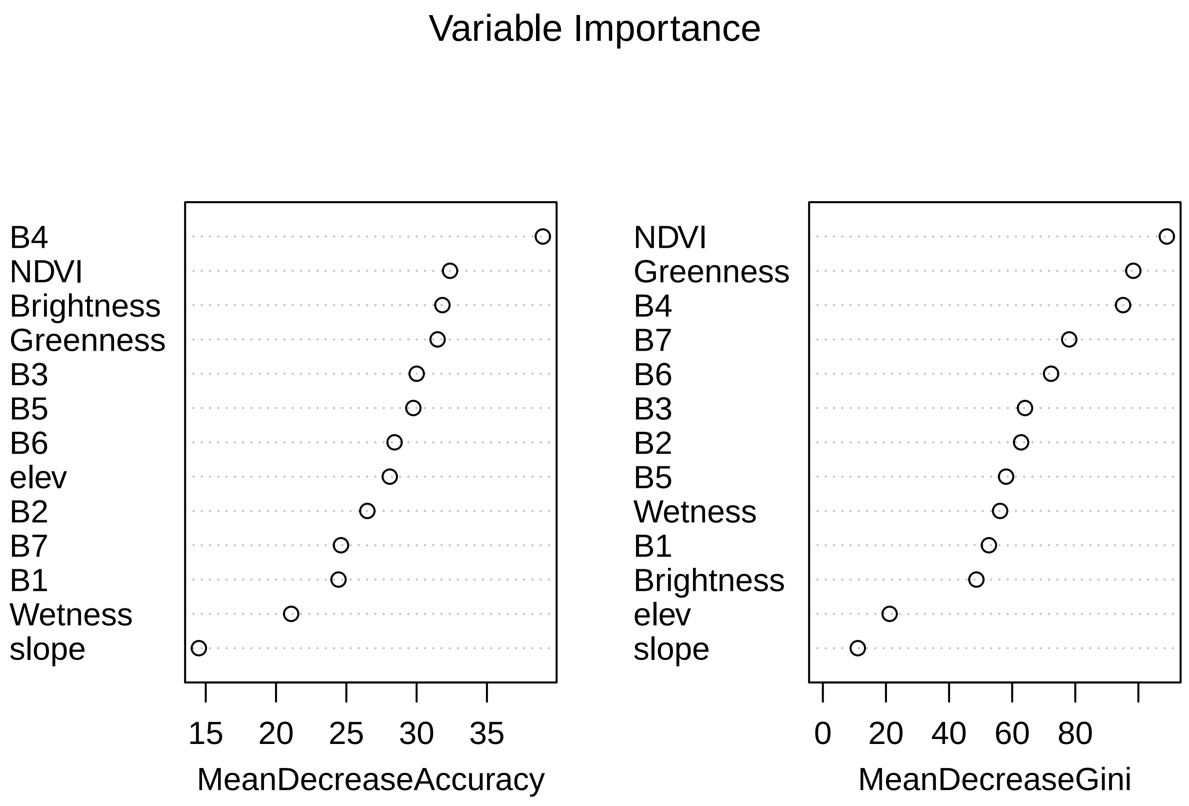
I used the caret library to create a function which would predict values for each pixel based on the random forest classification and then ran that functionw ith my data.

```
rf_predictions <- function(img, model_name){
  ##Create a raster with predictions from the fitted model project
  beginCluster()
  preds_rf <- clusterR(img, raster::predict, args = list(model = model_name))
  endCluster()
  return(preds_rf)
}

l8_2013217_preds <- rf_predictions(l8_2013217_rf_data, l8_2013217_rf)
```

I also created a plot of variable importance for each variable included in the random forest classification algorithm.

```
varImpPlot(l8_2013217_rf, main = "Variable Importance")
```



To get the confusion matrix values, I also needed to call the results of the random forest analysis.

```
l8_2013217_rf
```

```
##
## Call:
##  randomForest(formula = as.factor(class) ~ B1 + B2 + B3 + B4 +      B5 + B6 + B7 + elev + slope + ND
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
```

```
##
##         OOB estimate of  error rate: 6.09%
## Confusion matrix:
##    1   2  3  4  5  6  7   8  9 class.error
## 1 62   4  0  0  0  0  1   0  0  0.07462687
## 2  1 274  4  0  0  1  1   0  0  0.02491103
## 3  0   3 94  0  0  0  0   7  0  0.09615385
## 4  0   0  3 15  0  0  0   5  0  0.34782609
## 5  0   3  0  0 97  0  6   0  0  0.08490566
## 6  1   1  0  0  0 15  0   0  0  0.11764706
## 7  0   0  0  0  7  0 89   1  0  0.08247423
## 8  0   2  7  0  1  0  0 263  0  0.03663004
## 9  0   0  0  0  0  0  0   3 47  0.06000000
```

Finally, I wrote all of the rasters as GeoTIFFs so I could import them into ArcGIS and visualize them.

```
writeRaster(l8_2013217_preds, filename = "l8_2013217_model_predictions", format = "GTiff", overwrite = 
writeRaster(l8_2013217_tca, filename = "l8_2013217_tca", format = "GTiff", overwrite = TRUE)
writeRaster(l8_2013217_slope, filename = "l8_2013217_slope", format = "GTiff", overwrite = TRUE)
writeRaster(elev_utm22_resamp, filename = "l8_2013217_elev", format = "GTiff", overwrite = TRUE)
writeRaster(l8_2013217_ndvi, filename = "l8_2013217_ndvi", format = "GTiff", overwrite = TRUE)
```